جامعة حلوان
Helwan University

HELWAN UNIVERSITY
Faculty of Computing and Artificial Intelligence
Artificial Intelligence Department

SlideSpace : Automatic Slide Generation with Evolutionary Layout Optimization

A graduation project dissertation by:

Ahmed M. Makboul [20210**111**]

Ahmed Mostafa [20210**121**]

Hossam Ahmed [20210**281**]

Omar Ayman [20210**597**]

Ahmed Bassiouny [20210**098**]

Ahmed Ismail [20210**023**]

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computing & Artificial Intelligence at the Artificial Intelligence Department, the Faculty of Computing & Artificial Intelligence, Helwan University

Supervised by:

Dr. Amr S. Ghoneim

June 2025

جامعة حلوان
كلية الحاسبات والذكاء الاصطناعي
قسم الذكاء الاصطناعي

التوليد الآلي للعروض التقديمية مع التحسين التطوري لبنية وتنسيق التصميم

رسالة مشروع تخرج مقدمة من:

أحمد محمد مقبول [20210**111**]

أحمد مصطفي السيد [20210**121**]

حسام أحمد صلاح [20210**281**]

عمر أيمن فاروق [20210**597**]

أحمد محمد بسيوني [20210**098**]

أحمد إسماعيل سالم [20210**023**]

رسالة مقدمة ضمن متطلبات الحصول على درجة البكالوريوس في الحاسبات والذكاء الاصطناعي، بقسم الذكاء الاصطناعي ، كلية الحاسبات والذكاء الاصطناعي، جامعة حلوان

تحت إشراف

د. عمرو غنيم

يونيو/حزيران 2025

SlideSpace

Abstract

SlideSpace is an intelligent system designed to automate the creation of presentation slides by combining natural language processing with evolutionary layout optimization techniques. The platform aims to reduce the manual effort involved in designing visually compelling and logically structured slides from raw content such as documents, bullet points, or summaries. At its core, SlideSpace uses AI-driven content segmentation and formatting strategies to determine the most relevant information for each slide, while an evolutionary algorithm iteratively optimizes layout elements such as text placement, font sizes, color schemes, and image alignment for maximum clarity, balance, and aesthetic appeal. By simulating natural selection through operations like mutation and crossover, SlideSpace evolves slide designs over generations to identify the most effective visual presentation. This approach ensures that the resulting slides are not only content-rich but also professionally formatted and visually engaging, making SlideSpace a powerful tool for educators, professionals, and content creators seeking automated yet customizable slide generation.

Acknowledgement

We would like to thank everyone who supported us during the development of *SlideSpace*. We are especially grateful to our teachers and mentors for their helpful advice, support, and encouragement throughout this project.

We also sincerely thank the Alexandria Library cloud provider for giving us access to the computing resources we need. Their cloud platform helped us run our models and test our system effectively.

We are thankful to the open-source and research communities whose work on Retrieval-Augmented Generation (RAG), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Tau-optimization gave us ideas and tools that were very important for our project.

Finally, we appreciate the help of our friends who gave us useful feedback and helped us make the system better.

Table of Contents

Abstract 5

Acknowledgement 7

Table of Contents 9

List of figures 12

List of Tables 14

1 Introduction 16

2 Literature Review 17

2.1 About the problem 17

2.2 Related work 17

2.2.1 Research on Document-to-Presentation Conversion 17

2.2.2 Approaches to Automated Presentation Generation and Human-AI Collaboration 27

2.2.3 Summarization approaches 28

2.3 Evaluation metrics 30

3 The Proposed Solution 36

3.1 Solution Methodology 36

3.2 Functional/ Non-functional Requirements 36

3.2.1 Functional Requirements 36

3.2.2 Non-Functional Requirements 36

3.3 System Analysis & Design 37

3.3.1 Main Objectives 37

3.3.2 Target Users 37

3.3.3 Core Features 37

3.3.4 System Components 38

3.3.5 User Classes and Characteristics 39

3.3.6 Example Workflow 39

3.4 System Diagrams 39

3.4.1 Class Diagram 39

3.4.2 Use Case Diagram 39

3.4.3 Sequence Diagram 40

3.4.4 Activity Diagram 41

3.4.5 Database (ERD) 42

3.5 User Experience (UX) & User Interface (UI) 43

4 Slide generation pipeline 48

4.1 Document Extraction Phase 49

4.2 Chunking Strategy........50
4.2.1 Recursive Chunking Approach........50
4.2.2 Exclusion of Alternative Chunking Strategies........51
4.3 Embedding and Indexing........52
4.3.1 Purpose of CLIP........53
4.3.2 Embedding Workflow........53
4.3.3 Role of Embeddings in RAG........53
4.3.4 Alternative Embedding Strategy which is Image Captioning for Cross-Model Retrieval 54
4.4 Indexing and Retrieval with ChromaDB........55
4.4.1 Indexing in RAG........55
4.4.2 ChromaDB Integration in Our System........56
4.4.3 Query Construction from Document Titles........59
4.5 Response Generation........60
4.5.1 How Generation Works in Our Pipeline........61
4.5.2 Prompting Strategies........61
5 Evolutionary layout Optimization (ELO)........63
5.1 Problem Formulation: Slide Layout Optimization as an Evolutionary Problem........63
5.1.1 Problem Statement........63
5.1.2 Optimization Variables (Genes)........63
5.1.3 Why Evolutionary Algorithms?........64
5.2 Evolutionary Layout Optimization Algorithms........64
5.2.1 Genetic Algorithm (GA) for Layout Optimization........64
5.2.2 Particle Swarm Optimization (PSO) for Layout Optimization........67
5.2.3 Memetic Algorithm for Layout Optimization........69
5.2.4 Hybrid GA + τ-Extremal Optimization (Hybrid EO-GA) for Layout........71
5.3 Results Comparison........74
5.3.1 Performance Analysis by Slide Type........76
6 Experiments and evaluations........79
7 Conclusion........82
8 References........84

List of figures

FIGURE 2-1 AN OVERVIEW OF THE DOC2PPT ARCHITECTURE 18

FIGURE 2-2 D2S ARCHITECTURE 19

FIGURE 2-3 PAPER2SLIDES ARCHITECTURE 20

FIGURE 2-4 PRESENTATIONS ARE NOT ALWAYS LINEAR ARCHITECTURE 23

FIGURE 2-5 DOCPRES ARCHITECTURE 25

FIGURE 2-6 SLIDE4N FLOW 28

FIGURE 3-1 CLASS DIAGRAM 39

FIGURE 3-2 USE CASE DIAGRAM 40

FIGURE 3-3 SEQUENCE DIAGRAM 41

FIGURE 3-4 ACTIVITY DIAGRAM 42

FIGURE 3-5 ENTITY RELATIONSHIP DIAGRAM OF OUR SYSTEM 43

FIGURE 3-6 SNAPSHOT OF THE HOMEPAGE (UI) 43

FIGURE 3-7 FEATURE PAGE (UI) 44

FIGURE 3-8 UPLOAD A DOCUMENT (UI) 44

FIGURE 3-9 PRESENTATION PREVIEW PAGE (UI) 45

FIGURE 3-10 USER PROFILE (UI) 45

FIGURE 3-11 LOGIN-PAGE (UI) 46

FIGURE 3-12 SIGNUP PAGE (UI) 46

FIGURE 3-13 ADMIN PAGE (UI) 47

FIGURE 4-1 SLIDESPACE SLIDE CONTENT GENERATION ARCHITECTURE 49

FIGURE 4-2 VECTOR SPACE IMAGE TAKEN FROM QDRANT ARTICLE 56

FIGURE 4-3 ANN SEARCH (QDRANT ARTICLE) 58

FIGURE 4-4 RAG SUMMARIZATION 60

FIGURE 4-5 FEW-SHOT PROMPT EXAMPLE 62

FIGURE 4-6 CHAIN OF THOUGHT PROMPT 62

FIGURE 5-1 GENETIC ALGORITHM FOR EVOLUTIONARY LAYOUT OPTIMIZATION 65

FIGURE 5-2 PARTICLE SWARM OPTIMIZATION FOR EVOLUTIONARY LAYOUT OPTIMIZATION 67

FIGURE 5-3 MEMETIC ALGORITHM FOR EVOLUTIONARY LAYOUT OPTIMIZATION 70

FIGURE 5-4 HYBRID T-EO-GA FOR EVOLUTIONARY LAYOUT OPTIMIZATION 72

FIGURE 5-5 EXECUTION TIME VS SOLUTION QUALITY 74

FIGURE 5-6 EXECUTION TIME DISTRIBUTION BY ALGORITHM 75

FIGURE 5-7 CONVERGENCE SPEED VS STABILITY 76

FIGURE 5-8 ALGORITHM EFFICIENCY SCORE 76

FIGURE 5-9 PERFORMANCE ANALYSIS BY SLIDE TYPE 77

List of Tables

TABLE 1 COMPARISON BETWEEN EVOLUTIONARY ALGORITHM 74

1 Introduction

In academic and educational environments, preparing presentation slides is often a complex and time-intensive task. Lecturers, students, and researchers frequently need to convert complex documents such as research papers, reports, or course materials into concise and visually effective slide presentations. This process typically involves reading and interpreting lengthy content, identifying key points, and manually arranging them into a slide format, which can be both mentally demanding and time-consuming.

SlideSpace addresses this challenge by offering a fully automated solution that transforms documents (e.g., PDFs) into well-structured presentation slides (e.g., PPTX format) using advanced AI techniques. At the heart of SlideSpace is a **Retrieval-Augmented Generation (RAG)** model that leverages natural language processing (NLP) to extract and summarize key content from input documents. This ensures that the generated slides are not only content-accurate but also focused on the most relevant and impactful information.

To further enhance the visual quality and effectiveness of the slides, SlideSpace integrates a **Genetic Algorithm (GA), Particle Swarm Optimization (PSO)**, and **Tau-optimization** strategies for evolutionary layout optimization. This algorithm simulates natural selection to evolve slide designs over multiple iterations, optimizing elements such as text placement, font size, image alignment, and color harmony. The result is a set of slides that are not only information-rich but also professionally designed with minimal human effort. By automating both content generation and visual layout, SlideSpace significantly reduces the cognitive load and preparation time for academic presenters, enabling them to focus more on delivery and less on design.

2 Literature Review

2.1 About the problem

In many academic, corporate, and educational settings, professionals often need to convert detailed documents rich with text and image into concise, engaging presentation slides. This task is both time-consuming and cognitively demanding. It requires not only summarizing and condensing large volumes of information but also organizing the content logically and visually for maximum audience engagement. Key challenges include deciding what information to keep, how to represent it visually, maintaining consistency in style, and ensuring clarity. Additionally, integrating images appropriately either by selecting from existing content or generating new visuals adds another layer of complexity. Automating or assisting this process can significantly enhance productivity and communication quality.

2.2 Related work

2.2.1 Research on Document-to-Presentation Conversion

The DOC2PPT [1] system is designed to automatically generate slide decks from multimodal documents, which contain both textual content and figures. This system comprises three primary modules, each of which plays a crucial role in the slide generation process:

1. **Document Reader (DR):** The Document Reader is responsible for encoding the text and figures within the document. It extracts the relevant information from each section and converts both the sentences and visual content into representations that can be processed by subsequent modules
2. **Progress Tracker (PT):** The Progress Tracker monitors the progression of the slide generation task. It maintains pointers to the current input section and the corresponding output slide, ensuring the correct flow of the process. This module determines when the system should transition to the next section or slide, based on the progress made so far
3. **Object Placer (OP):** The Object Placer decides which content whether textual or visual from the current section should be placed on the slide. Additionally, it predicts the location and size of each object to be placed on the slide, ensuring an optimal and coherent layout for the presentation

An overview of the DOC2PPT architecture and the interactions between these modules is illustrated in **Figure 2-1**.

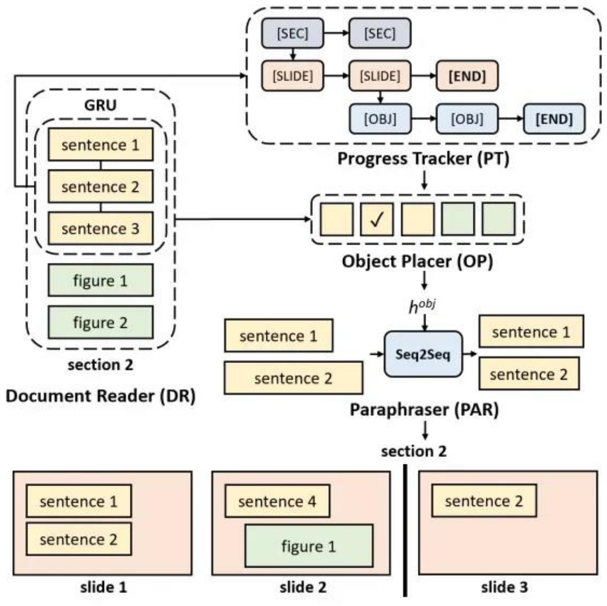


Figure 2-1 An overview of the DOC2PPT architecture

The **Paraphraser (PAR)** module is responsible for taking the selected sentence and rephrasing it into a more concise form suitable for slide presentation. This process ensures that the key information is preserved while eliminating unnecessary details, making it more digestible for the audience.

The **Document Reader (DR)** encodes the text by utilizing **RoBERTa** and a **bidirectional GRU** (Gated Recurrent Unit) to extract **contextualized sentence embeddings**. This approach, akin to the **attention mechanism**, enables the model to capture the semantic relationships between words in context, allowing for a more nuanced and accurate representation of the text.

The **ResNet-152** architecture, a convolutional neural network (CNN), is employed to extract the image embeddings for each figure in the document. Given a figure $F_q^{in} = \{I_q^{in}, C_q^{in}\}$, where I_q^{in} represents image content and C_q^{in} denotes the associated contextual information (such as captions or labels), the image features are extracted.

Subsequently, these extracted features are **concatenated** to form the **figure embedding V_q^{in}**, which captures both the visual and contextual information of the figure, enabling the system to represent the figure in a more comprehensive manner. This embedding serves as the foundational representation used in later stages of the presentation generation process.

$$F_q^{in} = \{I_q^{in}, C_q^{in}\}$$

$$V_q^{in} = I_q^{in} \oplus C_q^{in}$$

Before the development of **DOC2PPT**, the **D2S: Document-to-Slide Generation via Query-Based Text Summarization** approach [2](2021) proposed a **two-step method** to generate slides from documents. The first step involves **retrieving relevant content**, including text, figures, and tables, based on slide titles. In the second step, this content is **summarized** into concise **bullet points** using a **long-form question-answering model** designed for open-domain tasks.

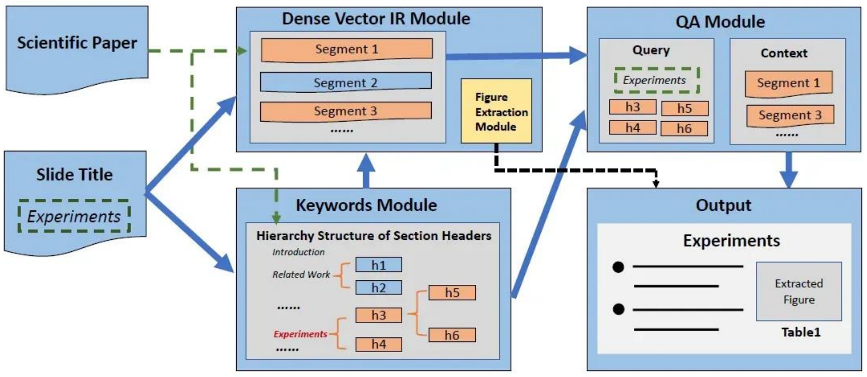


Figure 2-2 D2S architecture

The approach treats **user-centered slide titles** as queries and the document as the corpus, employing **information retrieval (IR)** to select the most relevant sections. These retrieved snippets are then passed to a QA model for **sequence-to-sequence generation**, resulting in the final slide content. Additionally, a **keyword module** is used to extract a hierarchical discourse structure from the paper, further enhancing the retrieval process by ranking the most relevant content for a given slide title.

The **Unsupervised Paper2Slides Generation** approach [3](2024) introduces a new technique for generating presentation slides from scientific papers, focusing primarily on **Natural Language Processing (NLP)** and **Natural Language Generation (NLG)** theories, specifically using **text summarization** and **text generation**. Unlike previous approaches, this

SlideSpace

system does not incorporate multimodal techniques and **does not utilize image information** from the document.

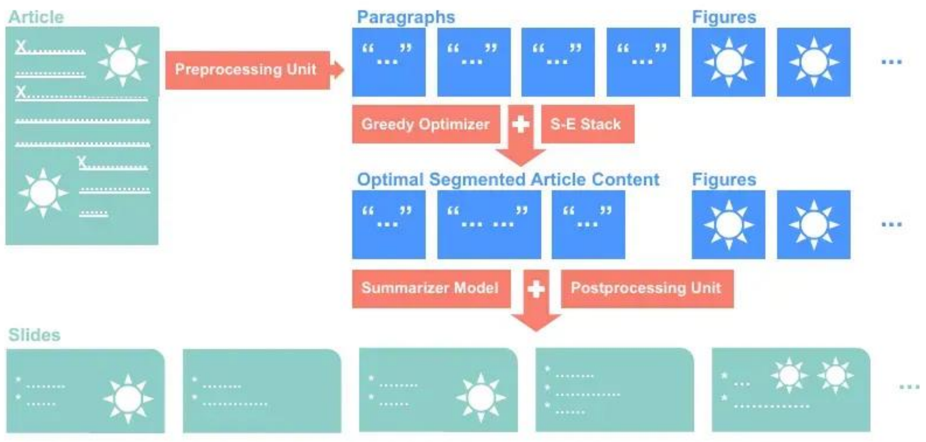


Figure 2-3 Paper2Slides architecture

The pipeline begins with the **Preprocessing Unit**, which extracts the text and identifies figures or images within the PDF document. The tool **GROBID** is employed to extract and process the text, cleaning it by removing irrelevant elements such as footers, headers, and tables of contents. GROBID is a machine learning library designed for parsing and restructuring raw documents (particularly scientific ones) into structured XML/TEI formats.

Additionally, the **PdfFigures2** tool is used to detect and extract figure-like objects such as charts, tables, and images from the PDF, providing the necessary visual content for slide creation.

After preprocessing, the **Model Generation Unit** uses a **Summarizer-Expander (S-E) stack**. This stack consists of two components:

1. **Summarizer**: A model that condenses the extracted text into summaries.

2. **Expander**: A model that performs the reverse task—expanding the summaries back into longer, more detailed versions, closely resembling the original text.

The optimization process, specifically Reconstruction Loss, ensures that no significant information is lost during summarization and expansion. The Back-Generation Step is an iterative process in which the summarizer and expander models are trained together. The summarizer generates summaries, while the expander reconstructs the original text from those summaries, and this cycle is repeated to improve both models' accuracy.

Once the text and figures have been processed, the Postprocessing Unit formats the content into the final presentation slides. The generated text and figures are transferred into a LaTeX template, which is then compiled into a PDF slide deck.

Finally, the Greedy Optimizer addresses the optimization problem of dividing the scientific paper into manageable segments, which serve as the input for the summarizer model.

Another recent contribution in the field is the work titled **"Automatic Slide Generation from Scientific Papers Based on Multimodal Learning"** [4] (2022/23), which proposes a dual-path summarization model that integrates both textual and visual information to produce high-quality slide content. This approach is particularly notable for its combination of extractive text summarization with image-text alignment, enabling a more informative and contextually grounded presentation format.

The textual component of the model follows a two-stage process. The first stage involves a **Ranking Module**, which identifies the most informative sentences from each section of a scientific paper. This step is essential, as processing every sentence in a paper would be computationally expensive. The module consists of several subcomponents. A **Title Representation Module** applies a BiLSTM with attention to the paper's title to capture key terms, while a **Content Representation Module** uses word-level BiLSTMs and attention layers to encode sentence and section embeddings. These representations are then combined to estimate the importance of each sentence and section using a feed-forward network. The model is trained using Binary Cross Entropy loss, with ROUGE-based F1 scores serving as guidance to match human-generated summaries.

In the second stage, the selected sentences are passed to an **Extractive Module** that employs a Heterogeneous Graph Attention Network (HGAT). This component enhances the selection of key content by modeling complex relationships across different textual units. Sentence embeddings are generated using two versions of the Longformer model one with a maximum sequence length of 512 and the other with 4096 to capture varying levels of detail. GloVe embeddings provide word-level representations, and a projection layer aligns all embeddings into a unified 300-dimensional space. A graph is then constructed with diverse edge types connecting words, sentences, and sections within and across document boundaries. This design enables the model to reason over both local and global structures in the paper, leading to a more accurate and context-aware summary. The final summary is constructed by selecting and ordering the most relevant sentences based on their original position in the document.

In addition to text summarization, the model incorporates a **Multimodality Module** to integrate figures and other visual elements into the slide content. This module relies on CLIP (Contrastive Language–Image Pre-training), a powerful model trained to compute similarity scores between textual and visual representations. By fine-tuning CLIP on image-caption pairs from scientific literature, the system improves its ability to accurately match sentences from the summary with the most relevant figures from the paper. This approach allows the final slide deck to present a more balanced and visually grounded representation of the original document.

One of the most compelling aspects of this approach is its modularity. The summarization and image-matching components are designed to function independently, allowing for easy updates or replacements. For example, a different summarization model could be integrated without altering the image alignment process. Compared to more integrated and complex systems like DOC2PPT, this model offers a flexible and scalable alternative, making it especially suitable for academic settings where adaptability and ease of maintenance are important.

This approach builds on DOC2PPT [1] and the discourse-aware summarization model in Summarizing Long-Form Document with Rich Discourse Information [5], combining their strengths while introducing key innovations. Unlike previous models that rely on GRUs or CNNs alongside BERT or RoBERTa, this work uses **Longformer** directly for sentence and section embeddings, enabling better handling of long documents. Most notably, it extends the summarization method from [5] into a **multimodal** setting by incorporating image content into both the summarization and slide generation process to make an advancement not explored in the earlier works.

The study titled *"Presentations are not always linear!"* [6] introduces a novel document-to-presentation generation framework that emphasizes the non-linear nature of presentation structure and the importance of content attribution. Unlike earlier works that primarily rely on sequential document-to-slide mappings, this approach treats the task as a graph-based paragraph clustering problem augmented by large language models for final slide generation.

The authors formalize the task using a dataset of paired documents and presentations, where each document is a sequence of paragraphs and each presentation consists of an unordered sequence of slides. The core goal is to generate a coherent presentation from a new document, with a user-defined number of slides, such that the resulting slide content is logically organized and properly attributed to its source paragraphs.

To achieve this, the method is divided into several components. First, a classifier is trained using document-presentation pairs from the SciDuet dataset to determine whether two paragraphs belong on the same slide. Sentence embeddings are used to compute cosine similarity between paragraphs and slides, and these similarities guide the creation of positive and negative training samples. A RoBERTa-based binary classifier is fine-tuned using a weighted loss to handle the class imbalance, achieving high performance in predicting paragraph-level slide associations.

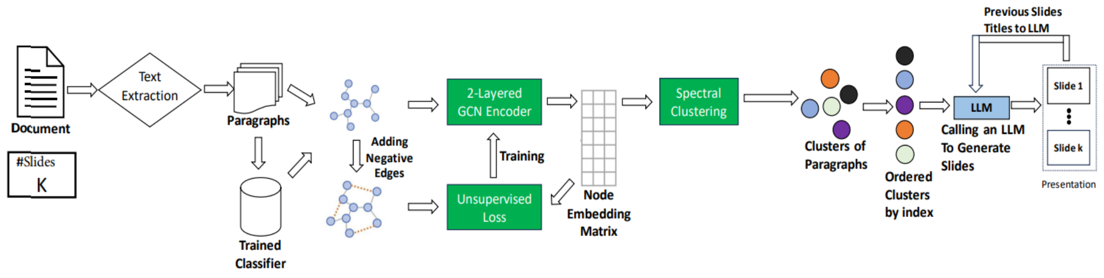


Figure 2-4 Presentations are not always linear architecture

Next, the model constructs a graph where each paragraph is a node and edges represent high-probability connections derived from the classifier. An unsupervised Graph Neural Network (GNN) is trained on this graph to learn paragraph embeddings. These embeddings are then clustered using spectral clustering, producing groups of paragraphs that are likely to form coherent slide content. Importantly, this graph-based clustering captures the non-linear flow of content better than traditional sequential models.

Following clustering, the model applies a simple heuristic to order the slide clusters based on paragraph position within the source document, maintaining a rough narrative flow. For slide content generation, GPT-3.5 is used to convert each paragraph cluster into slide text. The model receives not only the cluster's paragraphs but also the titles of preceding slides to ensure continuity and coherence across the generated presentation.

This work introduces several key contributions. It is among the first to explicitly model the **non-linear** nature of presentations through paragraph-level graph construction and clustering. Moreover, the integration of a **GNN-based structure learning module** with a **pretrained large language model** for natural language generation creates a flexible and powerful pipeline. Unlike previous approaches that assume linear mapping or rely solely on summarization techniques, this method prioritizes accurate content attribution and cohesive narrative structure across slides, offering a more user-centric and adaptable solution to presentation generation.

The paper *"Presentations Are Not Always Linear!"* [6] proposes a fundamentally different perspective on document-to-presentation generation by challenging the assumption of linear content flow. The central hypothesis is that effective presentations require a **non-linear structure** and precise **attribution** of slide content to specific parts of the source document. To achieve this, the authors introduce a two-stage framework: first, a **RoBERTa-based classifier** predicts the likelihood of paragraph pairs appearing on the same slide; second, these relationships are encoded as a graph, and a **Graph Neural Network** is trained to learn paragraph embeddings for **unsupervised clustering** into slide groups. Finally, **GPT-3.5** generates fluent slide content from each cluster, ensuring contextual coherence.

What sets this work apart from prior approaches such as DOC2PPT's [1] multimodal extraction or D2S's [2] query-driven summarization is its emphasis on **structural modeling** of intra-document relationships. Rather than aligning content to slides via section progression or title queries, it builds an adaptive graph that reflects **semantic cohesion** across the entire document. Moreover, unlike the previous multimodal methods that integrate images or rely on predefined summarization blocks, this method offers **flexibility in slide count**, **content granularity**, and **attribution clarity**, making it especially suited for dynamic and user-driven presentation creation.

"Enhancing Presentation Slide Generation by LLMs with a Multi-Staged End-to-End Approach" [7] introduces a structured pipeline for slide generation that combines hierarchical summarization, outline planning, and content-grounded generation through large language models (LLMs). The key hypothesis is that organizing content hierarchically and guiding LLMs with structured summaries can improve coherence, reduce hallucinations, and better reflect the document's intent.

The approach begins by extracting text and figures from the document and constructing a **bird's-eye view** a hierarchical summary of sections and sub-sections. This summary is used to guide LLM-based outline generation (slide titles), which are then mapped to the relevant document sections using fuzzy matching to ensure **semantic grounding**. Slide text is generated with awareness of prior slide content for narrative continuity. Finally, **CLIP-based vision-language matching** selects supporting images based on relevance to the generated slide text.

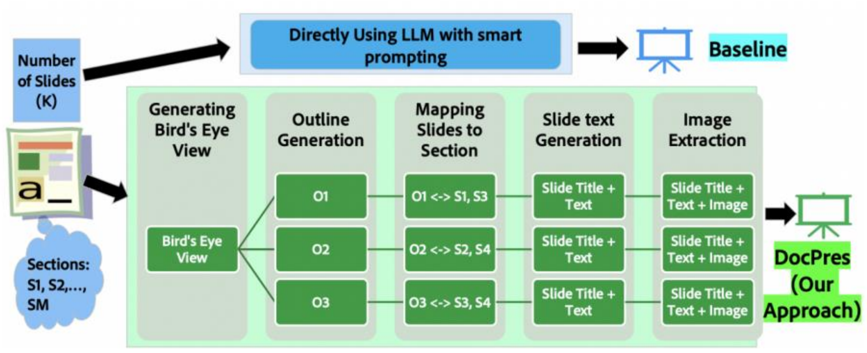


Figure 2-5 DocPres architecture

This method differs from prior work such as **DOC2PPT** [1] or **D2S** [2], which rely more directly on information retrieval or QA-based summarization. Unlike **graph-based approaches** [6] that explicitly model inter-paragraph relationships (e.g., GNN-based frameworks), this paper leans on LLMs to implicitly encode discourse structure. Its multi-stage architecture offers greater **modularity, controllability, and robustness**, especially for longer documents and varied user needs.

"SmartEDU: Accelerating slide deck production with Natural Language Processing" [8] presents a comprehensive platform designed to expedite slide deck creation from textual documents by integrating document preprocessing, automatic summarization, and slide composition.

The approach begins with **pre-processing**, where the system extracts structural elements such as titles, sections, and images from both structured and unstructured documents. For

PDFs, tools like **GROBID** and **PyMuPDF** enable accurate text and image extraction, preparing content for downstream processing.

Next, the **automatic summarization** phase leverages both extractive and abstractive methods, with supervised and unsupervised techniques available to flexibly condense content depending on data availability and task requirements. This ensures that essential information is succinctly captured for presentation.

Finally, **slide composition** organizes summaries into coherent slides. Structured documents benefit from directly using existing section titles as slide headings, while unstructured texts undergo **topic modeling** with LDA to infer thematic divisions. Where section titles are absent, trained models generate appropriate headings, facilitating clear organization. Users retain control, with the system producing draft slides open to manual refinement and enhancement.

To enrich visual content, SmartEDU integrates keyword and entity extraction (via **KeyBERT** and **spaCy**) to power an image retrieval mechanism sourcing relevant public domain images through a Bing crawler, thereby supporting engaging and contextually relevant slide visuals.

Overall, SmartEDU's modular pipeline marries classical NLP techniques with pragmatic engineering to reduce slide deck production time, offering a balanced blend of automation and user customization.

The automatic generation of presentation slides from scientific papers has been addressed through various approaches, each with different methods for handling textual and visual content. **DOC2PPT** [1] employs a multimodal framework, using deep learning techniques like RoBERTa for text and ResNet-152 for images, offering high-quality results but at the cost of computational complexity. **D2S** [2] simplifies the process with query-based text retrieval and summarization, focusing on text but lacking strong integration of visual elements. **Paper2Slides** [3] further streamlines this by using text-only summarization, providing a faster solution but with limited visual content.

Automatic Slide Generation from Scientific Papers Based on Multimodal Learning [4] combines text summarization with image-text alignment via CLIP, presenting a more balanced approach but introducing added complexity. **Presentations Are Not Always**

Linear! [6] challenges the traditional linear structure by using graph-based clustering to group content, offering more flexible slide organization but requiring careful handling of transitions. **DocPres** [7] uses hierarchical summarization and LLM-based content generation, focusing on coherence and structure but potentially struggling with long or complex documents. **SmartEDU** [8] offers a modular system that combines preprocessing, summarization, and image retrieval, allowing user control, though it may not perform well with highly technical content due to basic summarization.

Overall, these systems differ in their handling of **multimodal data**, **summarization techniques**, **organizational flexibility**, and **computational demands**, with each offering trade-offs between efficiency, quality, and complexity.

2.2.2 Approaches to Automated Presentation Generation and Human-AI Collaboration

The paper **Slide4N** [9] introduces an interactive system designed to assist data scientists in generating presentation slides from computational notebooks through human-AI collaboration. The system integrates backend algorithms for content extraction and organization with a front-end interface built as a JupyterLab extension. It operates in four key stages:

1. **Locate:** It identifies relevant cells in the notebook by calculating semantic coherence and overlapping attributes between code cells.
2. **Distill:** For each selected cell, it generates slide content, including titles and bullet points.
3. **Arrange:** The system organizes and places outputs like plots or tables from selected cells onto slides.
4. **Refine:** Users can manually refine the generated slides in the interface, allowing for further customization.

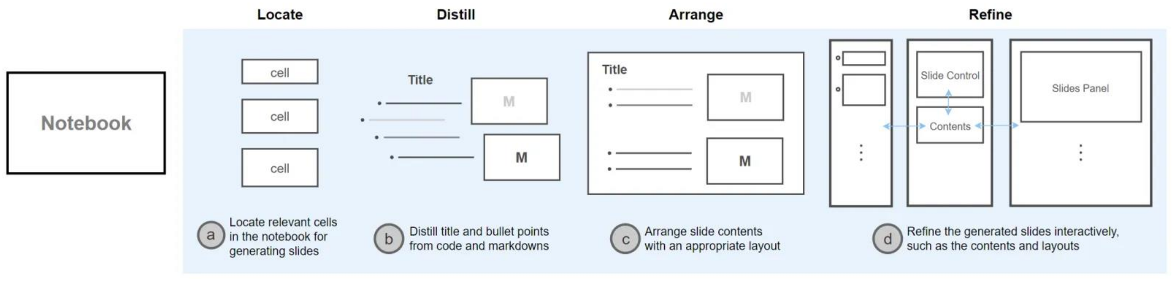

Locate
Distill
Arrange
Refine
Notebook
cell
cell
cell
Title
M
M
Title
M
M
Slide Control
Contents
Slides Panel
a Locate relevant cells in the notebook for generating slides
b Distill title and bullet points from code and markdowns
c Arrange slide contents with an appropriate layout
d Refine the generated slides interactively, such as the contents and layouts

Figure 2-6 Slide4N flow

This system emphasizes a collaborative approach, where AI aids in generating an initial slide deck, but the user retains the ability to refine the content, making it a flexible and efficient tool for data scientists.

2.2.3 Summarization approaches

The challenge of automatically generating presentation slides from scientific papers has been approached through diverse summarization techniques, each with distinct advantages. In Automatic Presentation Slide Generation Using LLMs [10], the authors employ a combination of Longformer-Encoder-Decoder (LED) and BigBird-Pegasus models, specifically fine-tuned on the PS5K dataset of paper-slide pairs. These transformer-based models demonstrate particular effectiveness in handling long document sequences while learning the concise, bullet-point style characteristic of presentation slides. The LED model's ability to process up to 16,834 tokens proves especially valuable in minimizing content truncation, a common limitation when working with lengthy scientific papers. This work also introduces an innovative use of DistilBERT embeddings to align sections of academic papers with their corresponding slides through cosine similarity calculations, creating a mapping between source content and slide content.

The Unsupervised Paper2Slides Generation paper [3] presents a markedly different approach through its Summarizer-Expander framework. Here, the authors leverage BART's sequence-to-sequence capabilities in a two-phase process that first condenses source material and then strategically expands the condensed points into slide-appropriate content. This method shows adeptness at handling varying information density, effectively distinguishing between core conceptual material and peripheral content. The paper provides compelling examples where the system successfully identifies and emphasizes key theoretical components, such as in its treatment of "bag-of-features" representations, while deemphasizing less critical implementation details. The unsupervised aspect of this approach offers notable advantages in terms of reduced reliance on large annotated datasets.

A graph-based methodology emerges in Automatic Slide Generation from Scientific Papers Based on Multimodal Learning [4], which represents documents as heterogeneous graphs incorporating words, sentences, and sections as nodes with multiple relationship types. The use of Graph Attention Networks (GATs) allows the model to capture hierarchical document structures and inter-sentence relationships that prove crucial for effective content extraction. This approach combines with BiLSTM networks and attention mechanisms to first score and then select the most salient content. The paper demonstrates how this method maintains better contextual flow compared to simpler extraction techniques, as it preserves relationships between selected sentences while still achieving the necessary compression for slide generation.

The SmartEDU [8] study provides perhaps the most comprehensive comparative analysis of summarization techniques for this task, evaluating everything from classical algorithms like TextRank and LexRank to advanced transformer models including Pegasus and DistilBART. Their findings reveal an important nuance - while supervised abstractive methods generally achieve higher performance metrics, unsupervised extractive methods show greater robustness across different domains and languages. This work also highlights the practical challenges of transformer-based approaches, particularly the sequence length limitations that necessitate document segmentation strategies. Interestingly, their results indicate that models like DistilBART, despite being trained on general domain data (CNN/DailyMail), can be effectively adapted to scientific slide generation through careful fine-tuning.

Scientific Papers Slide Generation Using Abstractive Text Summarization [11] offers valuable insights into the continuing evolution of these techniques, comparing traditional extractive methods like TF-IDF and Latent Semantic Analysis with modern abstractive approaches. The paper provides a particularly thorough examination of how different model architectures handle the unique requirements of scientific content, where precision and conceptual clarity are paramount. Their analysis of reconstruction loss as a metric for evaluating information retention during summarization proves especially insightful for assessing slide-generation systems.

The collective findings from these studies demonstrate that while transformer-based abstractive methods generally produce higher quality results, their computational demands and data requirements may make hybrid or extractive approaches more practical in many real-world scenarios. The research also consistently identifies document segmentation and content alignment as critical challenges, with various innovative solutions proposed across different papers. As the field progresses, the integration of these summarization techniques with layout generation and multimodal content processing appears poised to become the next major area of advancement in automated slide generation systems.

2.3 Evaluation metrics

To assess the performance and overall quality of the *SlideSpace* system, we use a combination of content-based, layout-based, optimization-based, and user-centered evaluation metrics. For clarity and completeness, we organize these metrics into four main categories: **Content Relevance**, **Layout Quality**, **Optimization Performance**, and **User Feedback**. Each category addresses a different aspect of the system, helping us measure how well *SlideSpace* extracts key information, produces readable and well-structured slides, and meets user expectations in real-world academic settings.

1- Content Relevance

This category focuses on how accurately and meaningfully the system extracts and summarizes information from input documents into slides. It checks how well the generated content matches the reference or ground-truth summaries.

Metric	Description
ROUGE-1 to ROUGE-4	Measures overlap of unigrams, bigrams, trigrams, and 4- grams between generated and reference summaries.

ROUGE-L	Measures the longest common subsequence (LCS) between the generated and reference summaries.
ROUGE-W	Weighted version of ROUGE-L that gives more credit to consecutive matches.
ROUGE-S	Measures skip-bigram overlaps (words in same order with gaps).
ROUGE-SU	Combines skip-bigram and unigram overlap.
ROUGE-SL	Adjusted ROUGE-L that considers slide length.
BERT Score	Uses BERT embeddings to compute similarity between tokens in generated and reference summaries.
BLEURT	Uses pretrained models to predict similarity based on human judgment.
G-Eval	Uses large language models (e.g., GPT) to judge generated text on coherence, fluency, relevance, and consistency.

1. ROUGE-N

   These metrics measure how many matching word sequences (called n-grams) appear in both the generated slide text and a reference summary. ROUGE-1 counts single words (unigrams), ROUGE-2 counts pairs of words (bigrams), ROUGE-3 counts triples (trigrams), and ROUGE-4 counts sequences of four words.

$$precision = \frac{Number\ of\ overlapping\ n-grams}{Total\ of\ n-grams\ generated\ in\ the\ summary}$$

   Example: If the reference summary says, “AI models are powerful” and the generated text says, “AI models are useful,” ROUGE-2 will recognize the bigram “AI models” as a match.

2. **ROUGE-L:**

   This measures the Longest Common Subsequence (LCS), or the longest sequence of words appearing in both texts in the same order, but not necessarily consecutively.

$$ROUGE-L = F1(LCS\ precision, LCS\ Recall)$$

where LCS Precision and Recall depend on the length of the LCS and total tokens in summaries.

Example: Comparing “the cat sat on the mat” and “the cat sat,” the LCS is “the cat sat.”

3. ROUGE-W : A weighted version of ROUGE-L that gives higher importance to consecutive matches rather than scattered matches. This helps better capture natural phrasing.

$$ROUGE - W = \frac{\sum_i w_i}{Length\ of\ summary},$$

$$where\ w_i\ is\ the\ wight\ of\ subsequences$$

Example: In "machine learning is fast" vs. "machine learning is efficient," the consecutive phrase "machine learning" receives more weight.

4. **ROUGE-S:**

   This metric counts skip-bigrams, pairs of words that appear in the same order but can have gaps between them.

$$ROUGE - S = \frac{Number\ of\ matching\ skip - bigrams}{Total\ skip - bigrams\ in\ reference\ and\ generated\ summaries}$$

   Example: "deep learning is good" and "deep models are good" share the skip-bigram "deep...good."

5. **ROUGE-SU:**

   An extension of ROUGE-S that adds unigram overlap, helping to capture both close and distant word relationships.

$$ROUGE - SU = ROUGE - S + ROUGE - 1$$

   Example: It detects both the skip-bigram "deep...good" and the unigram "learning."

6. **ROUGE-SL:**

   A modified ROUGE-L that adjusts for slide text length, ensuring summaries are concise yet informative.

$$ROUGE - SL = ROUGE - S \times e^{\frac{|Q - \bar{Q}|}{Q}}$$

   Use Case: Useful when slide content length varies greatly and needs penalizing or rewarding accordingly.

7. **BERT Score:**

   Instead of exact word matches, this metric uses BERT's contextual word

embeddings to measure semantic similarity between generated and reference text.
Example: “students love AI” and “learners enjoy artificial intelligence” score highly because their meanings are similar, even if words differ.

8. **BLEURT:**

   A learned metric based on pretrained transformer models fine-tuned on human annotations. It predicts how similarly generated text is to reference text, reflecting human judgment better than simple overlap metrics.
   Example: Paraphrased sentences with similar meaning receive higher BLEURT scores even if word overlap is low.

9. **G-Eval:**

   This uses large language models like GPT to evaluate text quality across four key dimensions: coherence, fluency, relevance, and consistency. It takes both generated and reference texts as input and outputs a detailed quality score.
   Example: GPT might score a generated summary as “Good fluency, some topic deviation” and assign a score 3.5 out of 5.

2-Layout Quality

This category evaluates the visual structure of the slides, including figure placement, text arrangement, and how well elements are aligned and organized.

Metric	Description
LC-FS (Longest Common Figure Subsequence)	Compares the sequence of figures in generated vs. ground-truth slides using LCS.
mIoU (Mean Intersection over Union)	Measures how well layout elements (text boxes, images) match ground-truth locations.
TFR (Text-Figure Relevance)	Evaluates whether figures are appropriately matched with their related text.

1-LC-FS (Longest Common Figure Subsequence):

This metric evaluates how well the order of figures (like images or charts) in the generated slides matches the order in the original (ground-truth) slides. It uses the Longest

Common Subsequence (LCS) method, which finds the longest sequence of figures appearing in the same order in both sets, even if they are not consecutive.

Example: If the ground-truth slide has figures [A, B, C, D] and the generated slide has [A, C, D], the LCS is [A, C, D], showing a good match except figure B is missing or reordered.

2-Mean Intersection over Union (mIoU)

mIoU measures how accurately the positions and sizes of layout elements (like text boxes and images) in generated slides overlap with their positions in ground-truth slides. It calculates the intersection area over the union area for each element and averages across all elements.

$$TFR = \frac{1}{M_F^{in}} \sum_{i=0}^{M_F^{in}} ROUGE(P_i, \tilde{P}_i)$$

Example: If a text box in the generated slide covers 80% of the same area as the reference text box, and partially overlaps but also extends beyond, mIoU quantifies this overlap as a score between 0 and 1, where 1 means perfect match.

3-TFR (Text-Figure Relevance):

This metric checks how well the text in a slide corresponds to the related figures. It uses a modified ROUGE score to measure the similarity between sentences referencing specific figures in both generated and ground-truth slides.

$$mIoU(O, \tilde{O}) = \frac{1}{N_O^{out}} \sum_{i=0}^{N_O^{out}-1} IoU(O_i, O_{J_i})$$

Example: If a generated slide describes “Figure 2 shows the sales increase” and the reference slide’s corresponding text also discusses sales increase linked to Figure 2, TFR will be high, showing good alignment between text and figures.

3 The Proposed Solution

3.1 Solution Methodology

SlideSpace was developed through a structured, scalable approach focused on addressing the challenges of summarizing and presenting scientific papers. It began with thorough requirement analysis to identify user needs, followed by designing intuitive UI/UX wireframes and implementing an MVC-based architecture. A well-structured relational database was created to manage users, documents, presentations, and interactions. The core functionality integrates an AI/NLP model to summarize papers into JSON-format slide content with titles, bullet points, and images. A dynamic front-end renders and allows editing of slides before export to PPTX or PDF. Additional features include a secure sharing system with view tracking, user authentication with role management, and feedback/rating mechanisms to continuously enhance presentation quality.

3.2 Functional/ Non-functional Requirements

The system is designed to deliver both functional capabilities that directly serve user tasks and non-functional qualities that ensure security, performance, and usability. Together, these requirements form the foundation for a reliable and user-friendly platform.

3.2.1 Functional Requirements

- Upload scientific papers (PDF/DOCX) with validation
- AI-based summarization into structured slide data (JSON)
- Slide editor with real-time preview and auto-save
- Export slides as PPTX or PDF with secure links
- Shareable presentation links
- Rating system (1–5 stars) with average rating display
- User accounts for managing files, presentations, and profiles
- Admin dashboard to monitor users, feedback, and analytics

3.2.2 **Non-Functional** Requirements

- Fast AI processing and response time
- Password hashing and input validation

• File sanitization for secure uploads
• Mobile-friendly, responsive interface
• Simple, intuitive UX for all user types
• Help/FAQ system for support
• High reliability and consistent uptime

3.3 System Analysis & Design

3.3.1 Main Objectives

The main objectives of **SlideSpace** are centered around simplifying academic content and enhancing presentation workflows for users in technical domains. The platform leverages AI to streamline the process from paper upload to slide export while promoting collaboration and feedback.

Main Objectives:

1. Simplify complex AI/ML/CV papers into short, clear slides.
2. Automate slide creation to save time for students and researchers.
3. Allow for real-time editing and exporting of presentation formats (PPTX/PDF).
4. Provide shareable links and downloadable formats for collaboration and delivery.
5. Facilitate feedback through a built-in presentation rating system.

3.3.2 Target Users

SlideSpace is designed to meet the needs of individuals involved in academic research, education, and presentations, particularly in technical domains. The key target users include:

• University students and professors in AI/ML/CV fields
• Researchers and PhD candidates
• Academic presenters and conference speakers

3.3.3 Core Features

SlideSpace is designed to streamline the presentation creation process with intelligent features that enhance user experience and productivity. Below are the key functionalities offered by the platform:

- Upload Paper
- AI-Based Summarization
- Slide Generation
- Slides Viewer
- Export Options
- Presentation Rating
- Shareable Link
- User Account System
- Admin Panel

3.3.4 System Components

The SlideSpace system architecture is organized into three main layers: Frontend, Backend, and the AI Model/Engine. Each layer handles specific tasks essential to the seamless transformation of uploaded scientific papers into presentation-ready slides.

1. **Frontend:**
   - Home page with clear CTA (upload, explore)
   - Upload Interface (Drag-and-drop + file preview)
   - Real-time Slide Viewer and Editor
   - Export/Download Controls
   - User Dashboard (Profile, Saved Presentations)
   - Admin Panel
2. **Backend:**
   - Document upload handler & metadata storage
   - Integration with Python model for AI generation
   - Slide data storage and export handler
   - Authentication & session management
   - Feedback and rating system
   - Admin control endpoints
3. **AI Model / Engine:**
   - NLP model to summarize and segment scientific papers
   - Outputs: JSON structure (title, bullets, layout hints)
   - Titles extraction
   - Image extraction from figures/tables in the paper

3.3.5 User Classes and Characteristics

- **Normal Users**: Upload papers, generate/view slides, rate, download, share slides.
- **Admins**: Manage users, monitor content/ratings, access statistics

3.3.6 Example Workflow

The following sequence illustrates how a typical user interacts with the SlideSpace platform—from uploading a paper to sharing a polished presentation.

1. The user uploads an AI paper (e.g., *"Vision Transformers: A Survey"*).
2. The AI model processes the document and summarizes the key sections (abstract, methodology, results).
3. Based on the summary, the system auto-generates a structured slide deck.
4. The user can review and edit the slides in real time, then export the presentation as PPTX or PDF.
5. Finally, the user can rate the presentation and optionally share it via a generated link

3.4 System Diagrams

3.4.1 Class Diagram

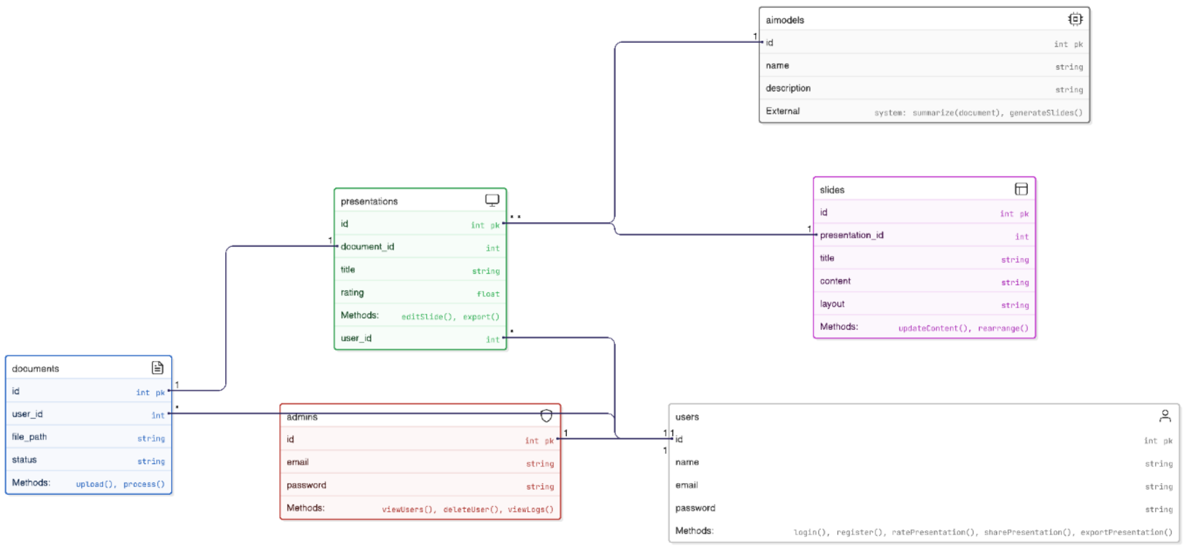


Figure 3-1 Class diagram

3.4.2 Use Case Diagram

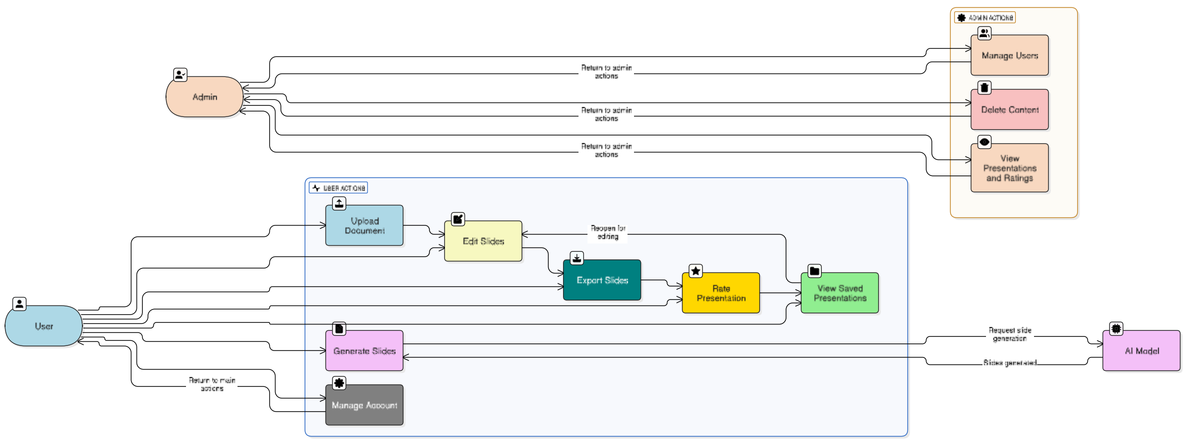
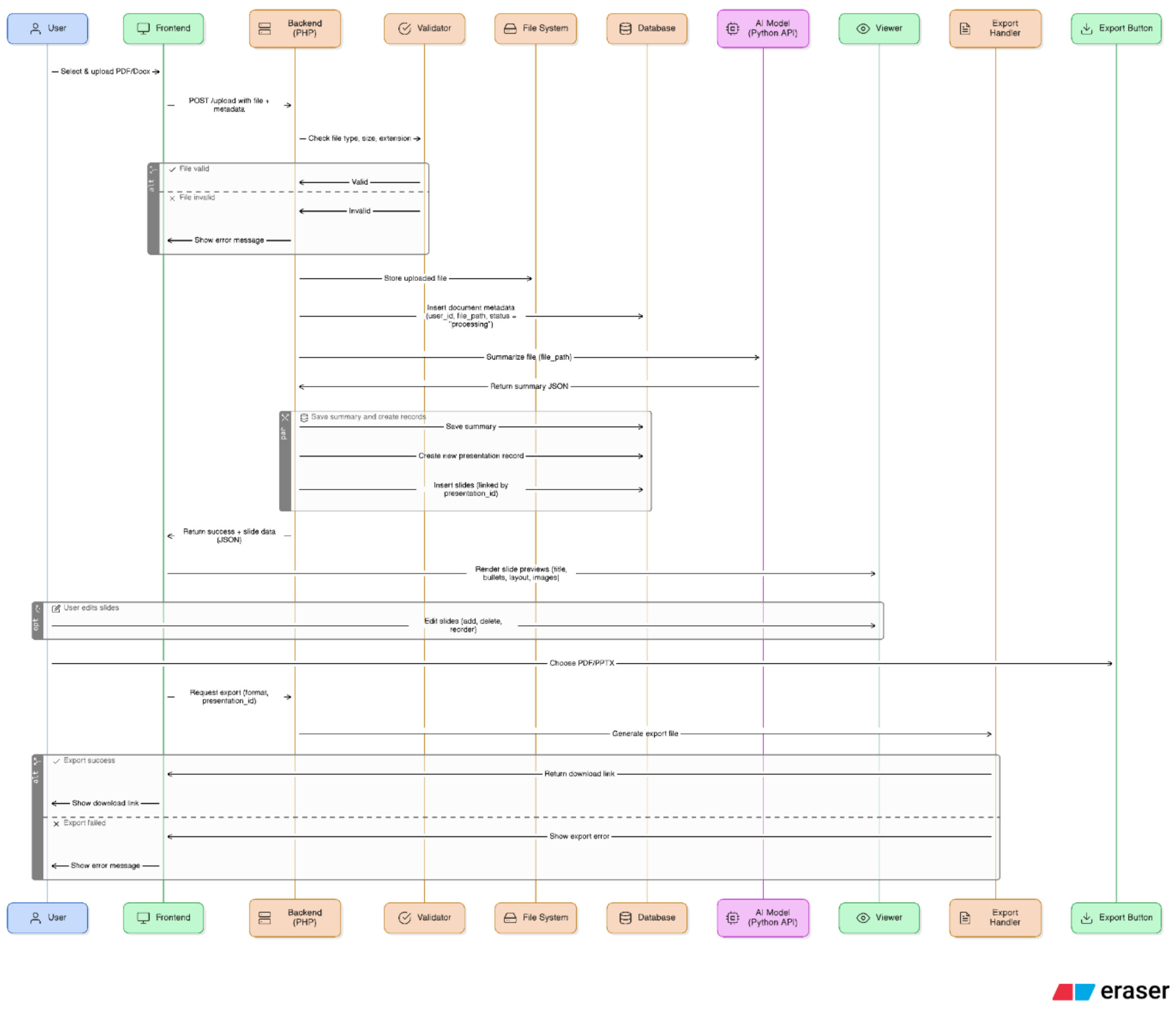


Figure 3-2 Use Case Diagram

3.4.3 Sequence Diagram



User
Frontend
Backend (PHP)
Validator
File System
Database
AI Model (Python API)
Viewer
Export Handler
Export Button
Select & upload PDF/Docx
POST /upload with file + metadata
Check file type, size, extension
alt
File valid
Valid
File invalid
Invalid
Show error message
Store uploaded file
Insert document metadata (user_id, file_path, status = "processing")
Summarize file (file_path)
Return summary JSON
par
Save summary and create records
Save summary
Create new presentation record
Insert slides (linked by presentation_id)
Return success + slide data (JSON)
Render slide previews (title, bullets, layout, images)
opt
User edits slides
Edit slides (add, delete, reorder)
Choose PDF/PPTX
Request export (format, presentation_id)
Generate export file
alt
Export success
Return download link
Show download link
Export failed
Show export error
Show error message
eraser

Figure 3-3 Sequence Diagram

3.4.4 Activity Diagram

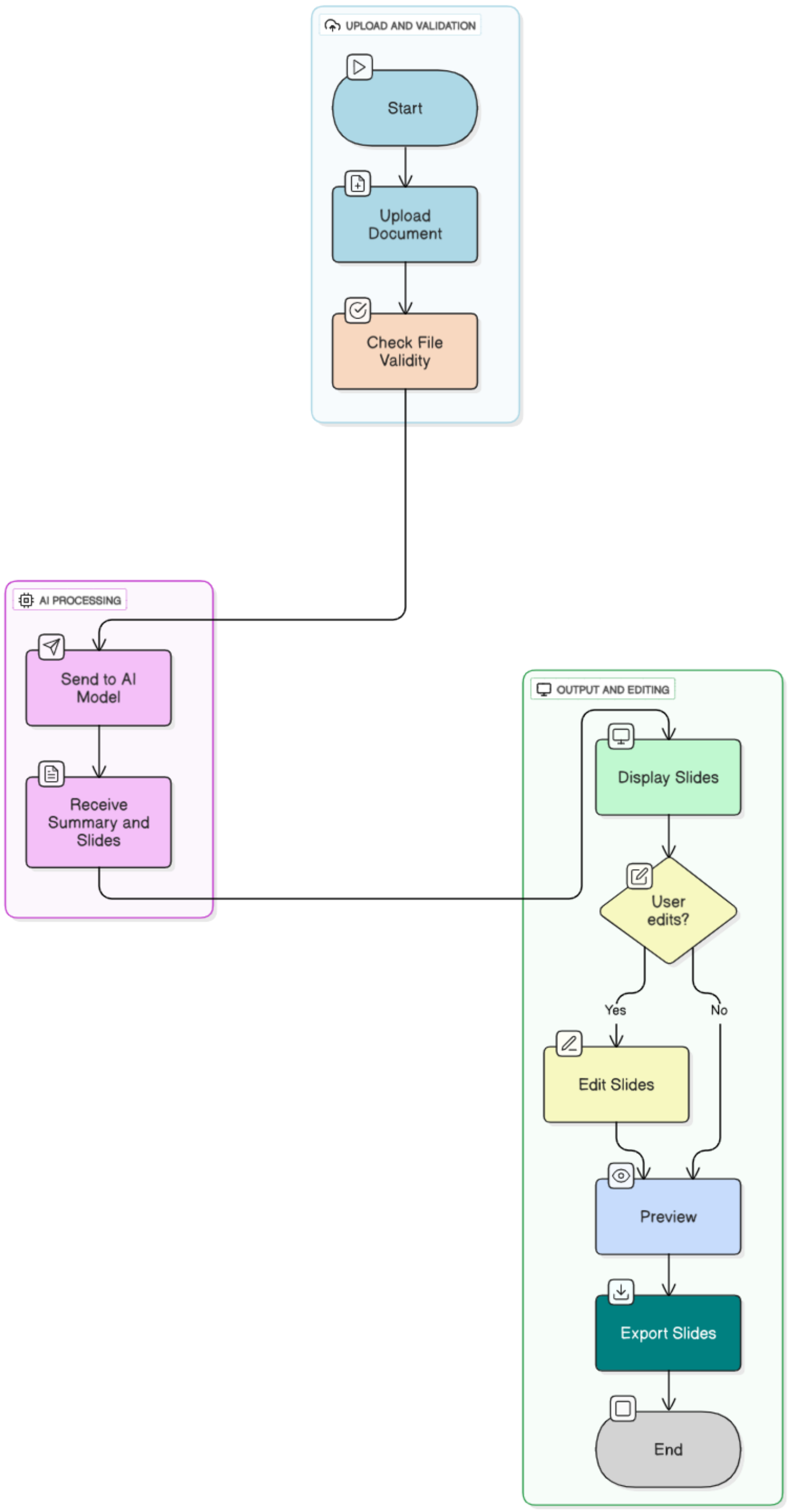


Figure 3-4 Activity Diagram

3.4.5 Database (ERD)

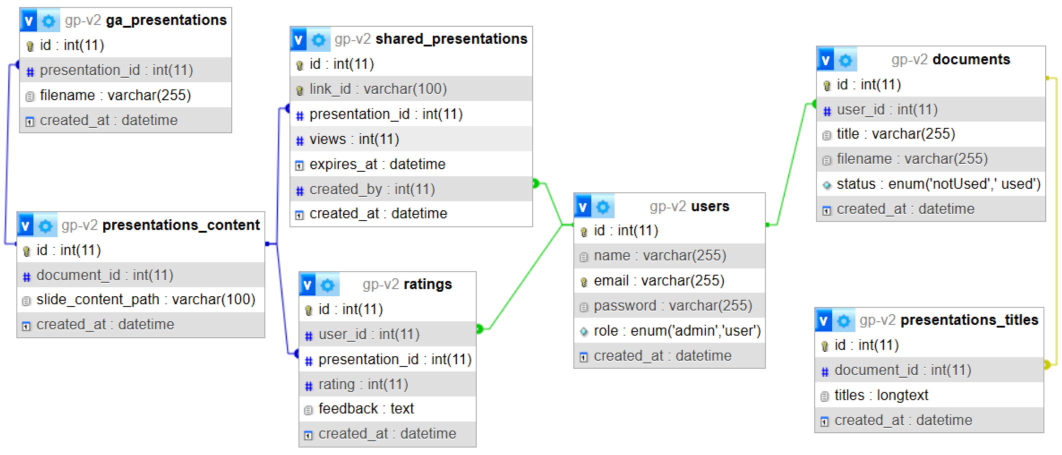
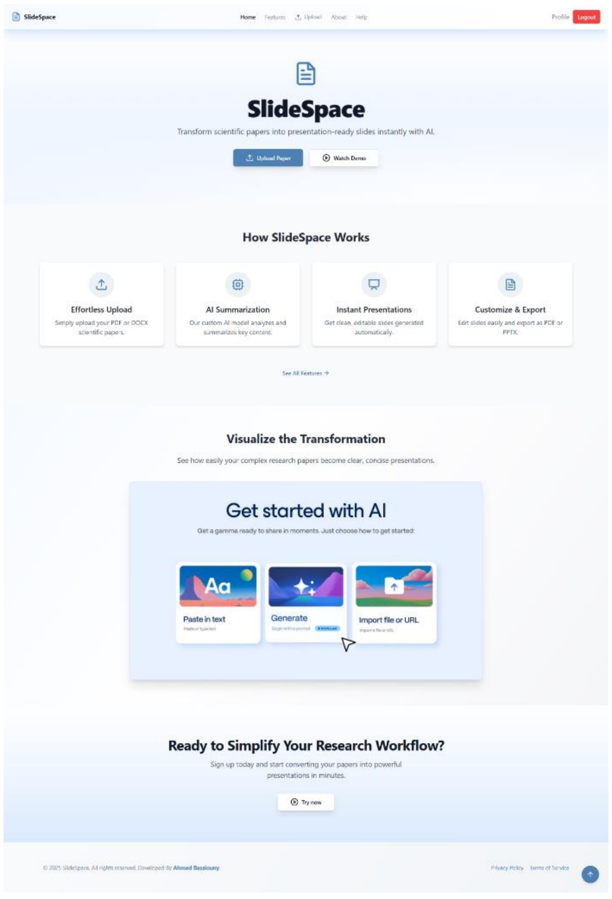


Figure 3-5 Entity relationship diagram of our system

3.5 User Experience (UX) & User Interface (UI)

1-Home page



SlideSpace

Transform scientific papers into presentation-ready slides instantly with AI.

How SlideSpace Works

Effortless Upload

Simply upload your PDF or DOCX scientific papers.

AI Summarization

Our custom AI model analyzes and summarizes key content.

Instant Presentations

Get clean, editable slides generated automatically.

Customize & Export

Edit slides easily and export as PDF or PPTX.

Visualize the Transformation

See how easily your complex research papers become clear, concise presentations.

Get started with AI

Get a gamma ready to share in moments. Just choose how to get started:

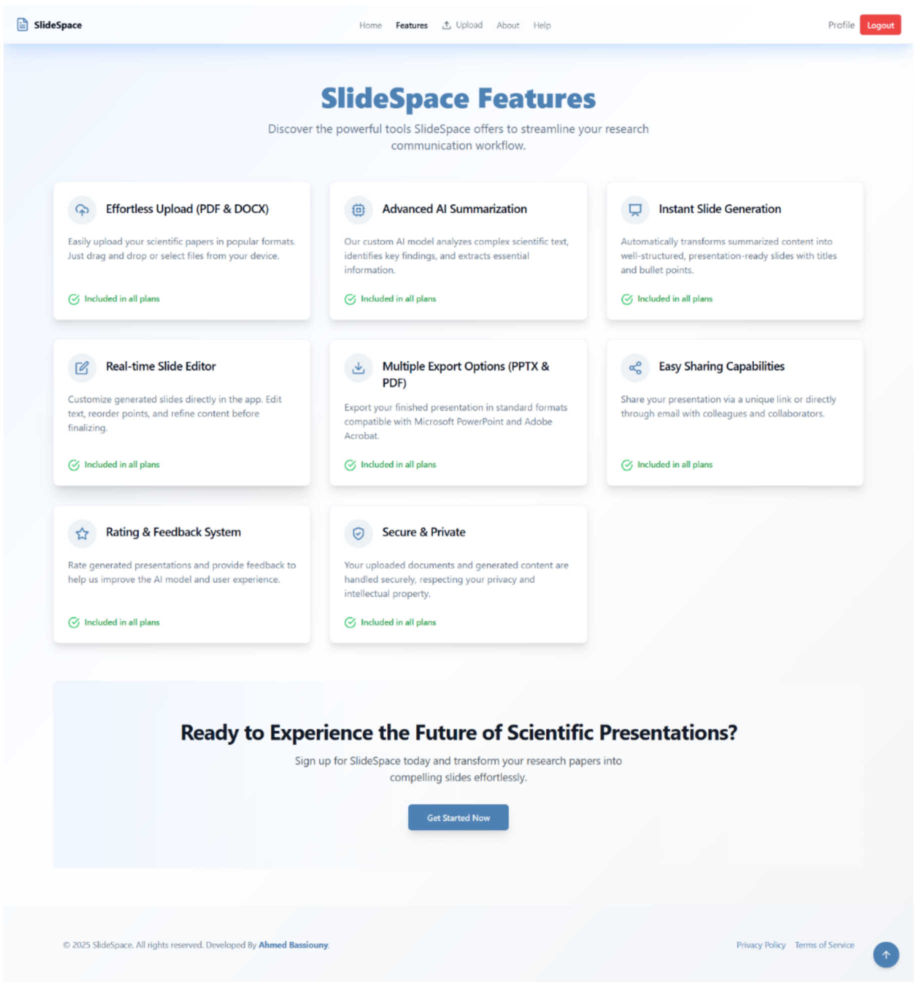
Ready to Simplify Your Research Workflow?

Sign up today and start converting your papers into powerful presentations in minutes.

© 2025 SlideSpace. All rights reserved. Developed By Ahmed Bassiouny.

Figure 3-6 Snapshot of the Homepage (UI)

2-Features page



SlideSpace Features

Discover the powerful tools SlideSpace offers to streamline your research communication workflow.

Effortless Upload (PDF & DOCX)

Easily upload your scientific papers in popular formats. Just drag and drop or select files from your device.

Included in all plans

Advanced AI Summarization

Our custom AI model analyzes complex scientific text, identifies key findings, and extracts essential information.

Included in all plans

Instant Slide Generation

Automatically transforms summarized content into well-structured, presentation-ready slides with titles and bullet points.

Included in all plans

Real-time Slide Editor

Customize generated slides directly in the app. Edit text, reorder points, and refine content before finalizing.

Included in all plans

Multiple Export Options (PPTX & PDF)

Export your finished presentation in standard formats compatible with Microsoft PowerPoint and Adobe Acrobat.

Included in all plans

Easy Sharing Capabilities

Share your presentation via a unique link or directly through email with colleagues and collaborators.

Included in all plans

Rating & Feedback System

Rate generated presentations and provide feedback to help us improve the AI model and user experience.

Included in all plans

Secure & Private

Your uploaded documents and generated content are handled securely, respecting your privacy and intellectual property.

Included in all plans

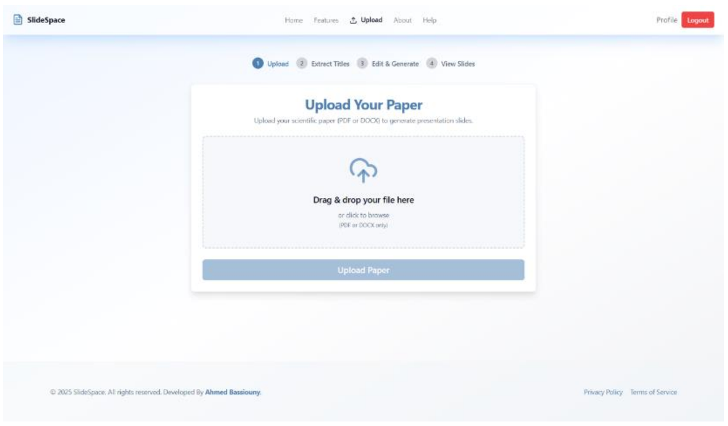
Ready to Experience the Future of Scientific Presentations?

Sign up for SlideSpace today and transform your research papers into compelling slides effortlessly.

© 2025 SlideSpace. All rights reserved. Developed By **Ahmed Bassiouny**.

Figure 3-7 Feature page (UI)

2- Upload page



Upload Your Paper

Upload your scientific paper (PDF or DOCX) to generate presentation slides.

Drag & drop your file here

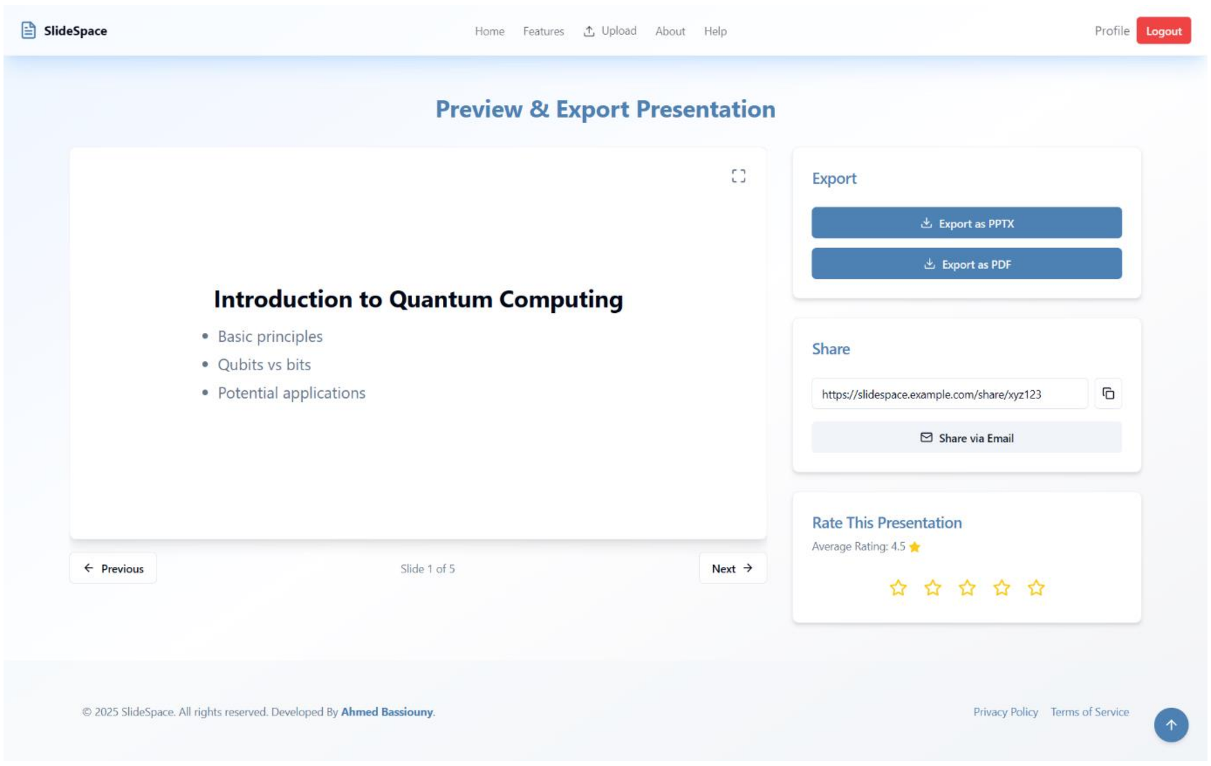
or click to browse

(PDF or DOCX only)

© 2025 SlideSpace. All rights reserved. Developed By **Ahmed Bassiouny**.

Figure 3-8 Upload a document (UI)

4-Preview page



Preview & Export Presentation

Introduction to Quantum Computing

- Basic principles
- Qubits vs bits
- Potential applications

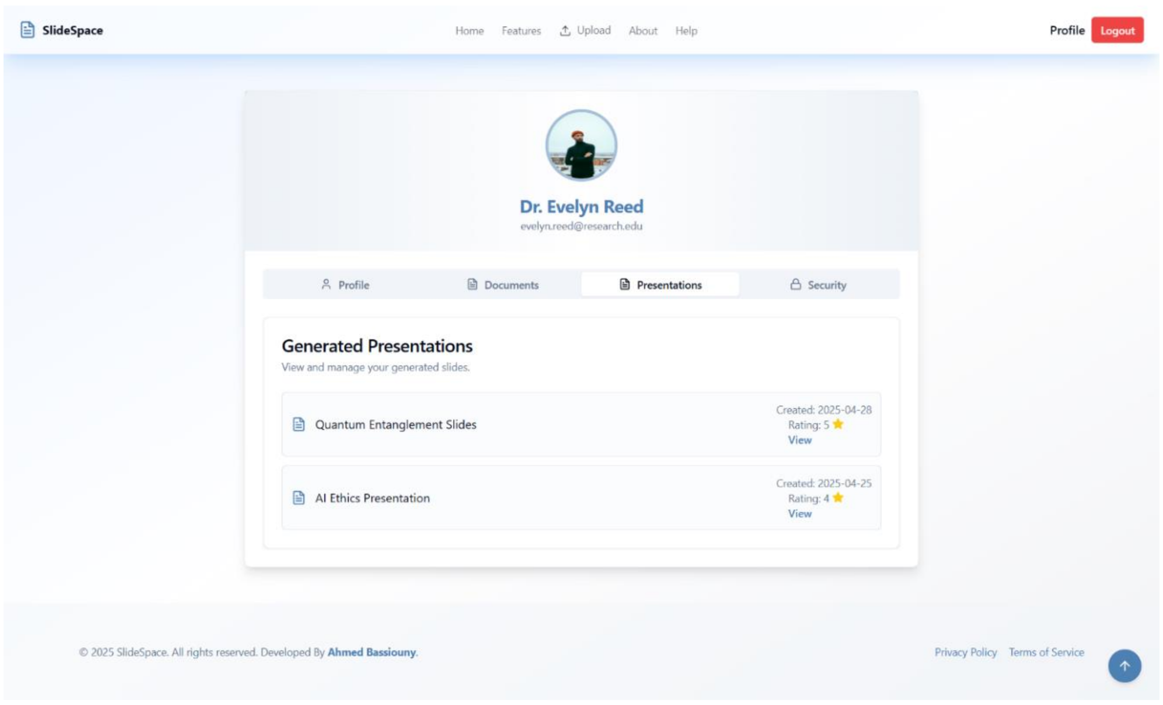
https://slidespace.example.com/share/xyz123

Average Rating: 4.5

© 2025 SlideSpace. All rights reserved. Developed By **Ahmed Bassiouny**.

Figure 3-9 Presentation preview page (UI)

5-Profile page



Dr. Evelyn Reed
evelyn.reed@research.edu

Generated Presentations

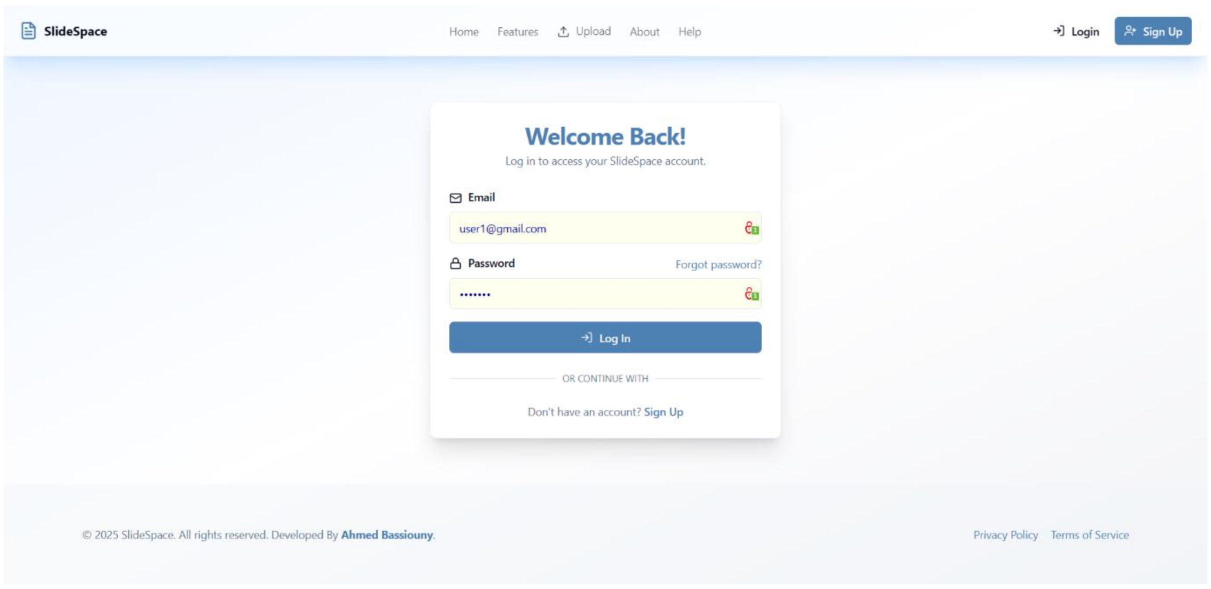
View and manage your generated slides.

Quantum Entanglement Slides	Created: 2025-04-28 Rating: 5 View
AI Ethics Presentation	Created: 2025-04-25 Rating: 4 View

© 2025 SlideSpace. All rights reserved. Developed By **Ahmed Bassiouny**.

Figure 3-10 User profile (UI)

6-Login page

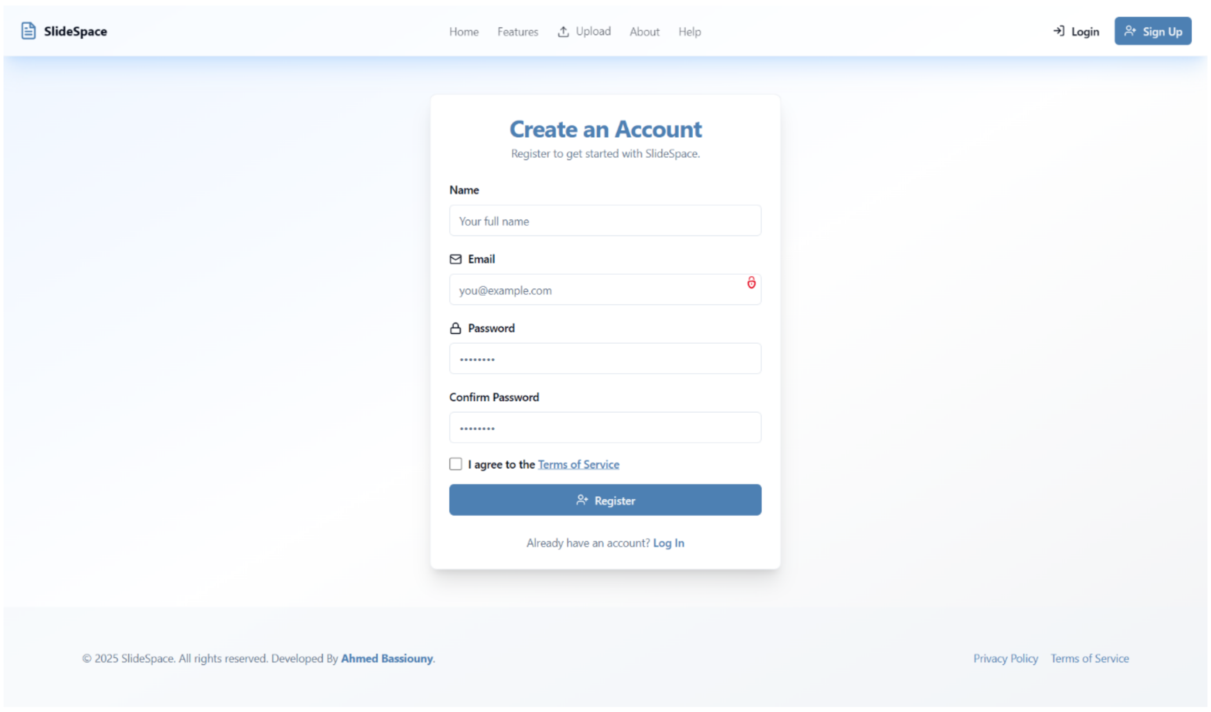

SlideSpace
Home Features Upload About Help
Login Sign Up

Welcome Back!
Log in to access your SlideSpace account.
Email
user1@gmail.com
Password Forgot password?
•••••••
Log In
OR CONTINUE WITH
Don't have an account? Sign Up

© 2025 SlideSpace. All rights reserved. Developed By **Ahmed Bassiouny**.
Privacy Policy Terms of Service

Figure 3-11 Login-page (UI)

6-Sign Up

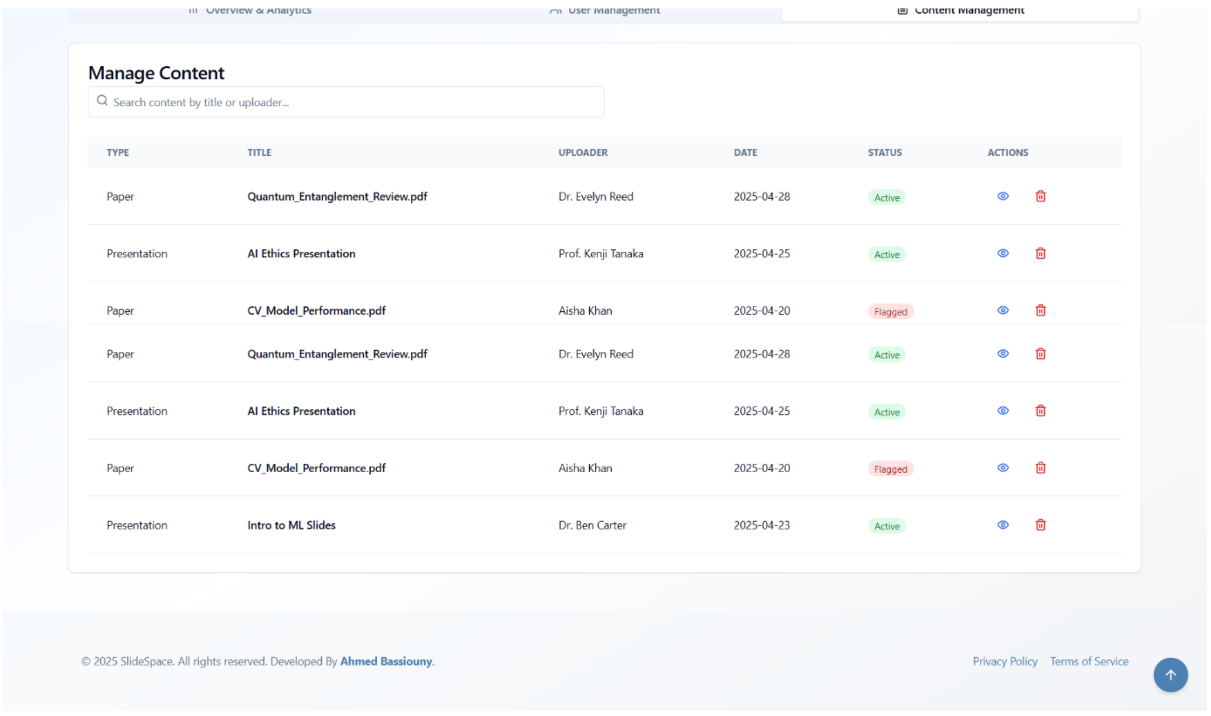

SlideSpace
Home Features Upload About Help
Login Sign Up

Create an Account
Register to get started with SlideSpace.
Name
Your full name
Email
you@example.com
Password
••••••••
Confirm Password
••••••••
I agree to the Terms of Service
Register
Already have an account? Log In

© 2025 SlideSpace. All rights reserved. Developed By **Ahmed Bassiouny**.
Privacy Policy Terms of Service

Figure 3-12 Signup page (UI)

7- Admin page



Overview & Analytics | User Management | Content Management

Manage Content

Search content by title or uploader...

TYPE	TITLE	UPLOADER	DATE	STATUS	ACTIONS
Paper	Quantum_Entanglement_Review.pdf	Dr. Evelyn Reed	2025-04-28	Active	
Presentation	AI Ethics Presentation	Prof. Kenji Tanaka	2025-04-25	Active	
Paper	CV_Model_Performance.pdf	Aisha Khan	2025-04-20	Flagged	
Paper	Quantum_Entanglement_Review.pdf	Dr. Evelyn Reed	2025-04-28	Active	
Presentation	AI Ethics Presentation	Prof. Kenji Tanaka	2025-04-25	Active	
Paper	CV_Model_Performance.pdf	Aisha Khan	2025-04-20	Flagged	
Presentation	Intro to ML Slides	Dr. Ben Carter	2025-04-23	Active	

© 2025 SlideSpace. All rights reserved. Developed By **Ahmed Bassiouny**.

Privacy Policy Terms of Service

Figure 3-13 Admin page (UI)

4 Slide generation pipeline

Retrieval-Augmented Generation (RAG) is a hybrid architecture that combines the strengths of information retrieval and generative language models. Introduced by Facebook AI, RAG integrates an external knowledge source (such as a document database or vector store) into the generation process by retrieving relevant documents in real-time and feeding them into a generative model like BART or T5.

In the context of summarization, RAG enhances the traditional capabilities of generative models by grounding the summary in retrieved, contextually relevant content. This approach is especially useful when dealing with large corpora or dynamic information that cannot be fully memorized by the model. By retrieving supporting information on-the-fly, RAG allows for more accurate, up-to-date, and context-aware summaries. It is particularly effective in domains like legal, scientific, or technical summarization, where precision and relevance are critical.

RAG-based summarization systems typically follow a two-step pipeline: (1) a retriever component identifies the most relevant documents or passages related to the input query or text, and (2) a generator composes a coherent and concise summary based on both the input and the retrieved context. This fusion of retrieval and generation allows for enhanced factual correctness and deeper understanding compared to standard abstractive summarization approaches.

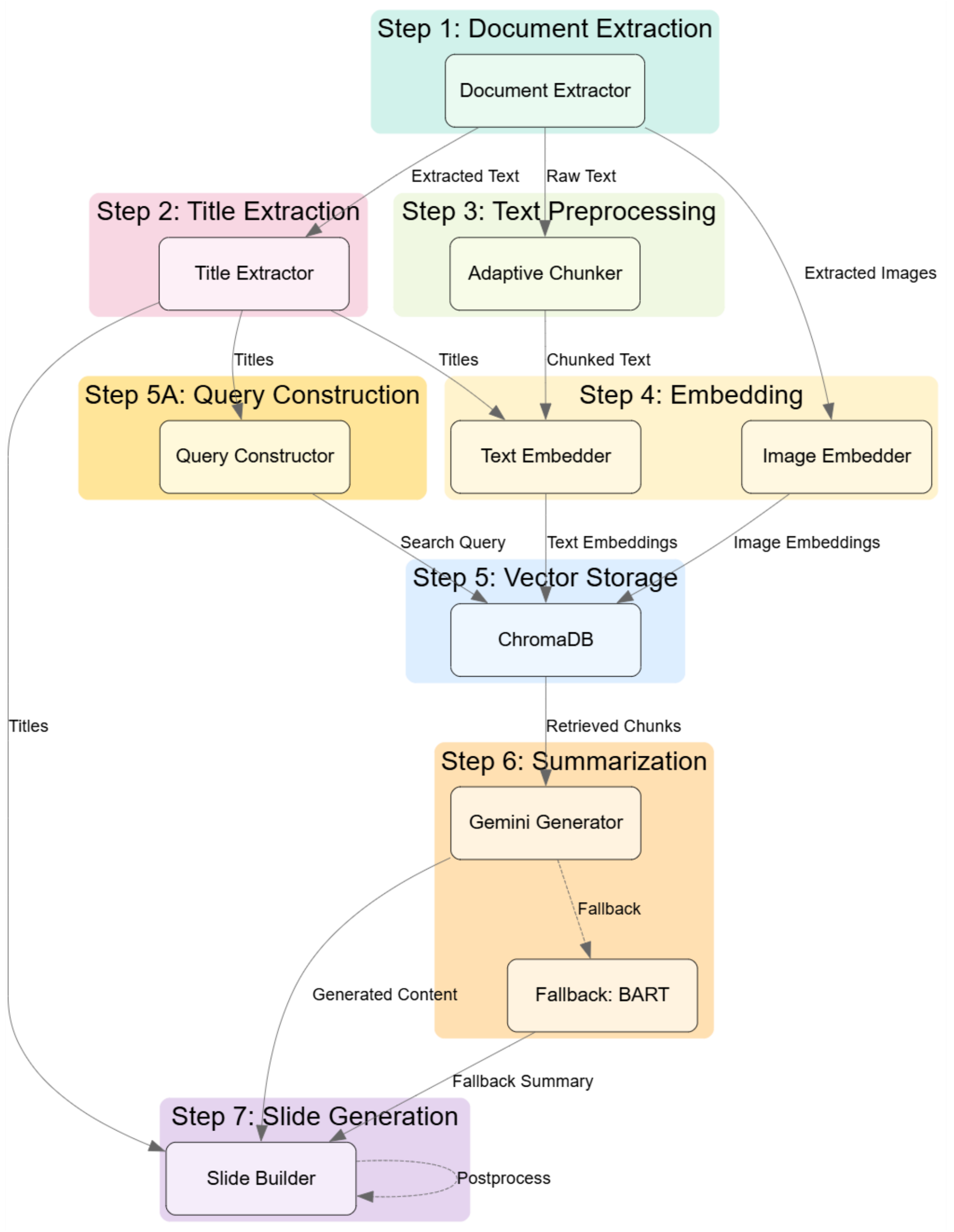


Figure 4-1 SlideSpace slide content generation architecture

4.1 Document Extraction Phase

The extraction phase serves as the foundational step in the Retrieval-Augmented Generation (RAG) pipeline, where raw data is systematically gathered from various document formats. In this project, we focused on two primary types of source documents: PDF and DOCX. To handle these formats, we utilized **PyMuPDF**—a lightweight, high-performance library capable of extracting structured and unstructured content from PDF files. Additionally, support for DOCX files was implemented to extend the versatility of the pipeline across common office document formats.

Using PyMuPDF, we extracted multiple content modalities, including **plain text**, **images**, and **tables**, thereby ensuring comprehensive data retrieval from complex document layouts. This multimodal extraction enhances the retriever's ability to match relevant content during the subsequent retrieval phase.

To support traceability and improve downstream processing, we enriched the extracted data with metadata such as **document index**, **page number**, and **section location**. These metadata tags provide crucial contextual signals, allowing the system to later calculate **relevance scores** and associate generated summaries with specific source locations—an important factor for explainability and verification in applications that demand high factual accuracy.

By systematically processing PDFs and DOCX files into a structured, metadata-enriched corpus, this extraction phase lays a robust groundwork for efficient and accurate information retrieval in the RAG-based summarization pipeline.

4.2 Chunking Strategy

After the initial extraction of raw content from PDF and DOCX documents, the next critical step in the pipeline is **text chunking**. Chunking refers to the process of dividing lengthy documents into manageable, context-preserving segments that can be effectively indexed and retrieved during the RAG process.

4.2.1 Recursive Chunking Approach

In our system, we adopt a **recursive chunking strategy**, which intelligently splits text based on a prioritized list of natural language boundaries—such as paragraphs, sentences, and words. This method allows the chunking process to maintain **semantic coherence** while still respecting a predefined token limit suitable for embedding models or language model inputs.

Compared to traditional **fixed-size chunking**, which divides text strictly by token or character count without regard for structure, recursive chunking preserves contextual integrity and reduces the likelihood of cutting off sentences or paragraphs mid-thought. This leads to higher quality embeddings and more relevant retrieval results.

Advantages of Recursive Chunking:

- Preserves logical boundaries (e.g., paragraphs, sentences).
- Produces more semantically coherent chunks.
- Reduces token waste by maximizing usable context within model limits.
- Balances efficiency and contextual fidelity better than rigid fixed-size splitting.

4.2.2 Exclusion of Alternative Chunking Strategies

Several other chunking paradigms **semantic-based**, **agentic-based**, and **document-based chunking**—were evaluated and ultimately excluded from our implementation due to practical limitations.

- **Semantic Chunking**: This method relies on dynamic segmentation based on topic shifts or embedding similarity. While promising in theory, it introduces **computational overhead** and **inconsistencies** in chunk length, leading to unpredictable performance and difficulties in indexing. Additionally, its effectiveness is highly sensitive to threshold tuning, which may not generalize well across diverse document types.

- **Agentic Chunking**: Agentic or agent-based chunking attempts to infer task-based boundaries or agent-intended sections within text (e.g., steps, instructions). However, this strategy is **domain-specific** and not broadly applicable to general documents like research papers or reports, where the "agent" is not always clearly defined.

- **Document-Based Chunking**: This technique treats each document as a single chunk. While it preserves full context, it is impractical for large documents that exceed model input limits. It also reduces retrieval granularity, leading to lower recall and reduced relevance when querying specific information.

Strategy	**Description**	**Advantages**	**Disadvantages**
Fixed-size chunking	Split text into fixed-length segments based on tokens or character count	- simple implementation - consistent chunk size	- Cuts across sentence boundaries - Low semantic coherence
Recursive chunking	Splits text at logical boundaries (e.g. paragraph , sentence, word) recursively	- Preserve structure and meaning - Efficient use of token space - Balanced chunks	- slightly more complex to implement (act as middle ground)
Semantic chunking	Divides text based on semantic similarity or topics shifts	- high contextual relevance - Adaptive to content	- Requires embedding & threshold tuning - Computationally expensive
Agentic chunking	Segments text by inferred agent steps or intensions	- Useful in procedural/instructional texts	- Domain-specific - Not generalizable to all document types

All of this make recursive chunking offers an effective middle ground—retaining structural meaning without sacrificing retrieval efficiency or requiring domain-specific logic. This makes it the most suitable choice for our RAG-based summarization system, particularly when dealing with heterogeneous document types and varying lengths.

We also apply table chunking where tables are parsed and split row-wise, treating each row as a semantically distinct unit which means that each row is transformed into a natural language sentence using column headers as context.

It helps in Retaining structured semantics of tabular data. Allows indexing and retrieval of meaningful table rows. Enhances LLM understanding when queried with natural language.

4.3 Embedding and Indexing

The third phase of our pipeline focuses on embedding the extracted and chunked content into a high-dimensional vector space, enabling efficient retrieval during the generation phase. For this task, we adopted the **CLIP (Contrastive Language–Image Pretraining)** [12] model, developed by OpenAI, which is capable of encoding both **text and images** into a shared embedding space.

This embedding phase ensures that retrieval is **semantic** (not keyword-based), **context-aware**, and **cross-modal**, which directly improves the quality and relevance of the generated output in RAG systems.

4.3.1 Purpose of CLIP

Unlike traditional language-only encoders (e.g., BERT or Sentence-BERT), CLIP provides **multi-modal embeddings**. This allows us to:

- Embed **textual content** (e.g., paragraphs, captions, or headings).
- Embed **visual content** (e.g., diagrams, figures, or screenshots).
- Store both modalities in the **same vector space**, enabling **cross-modal retrieval**.

This is particularly beneficial for documents such as research papers, reports, or technical manuals, where the meaning often depends on a combination of text and images.

4.3.2 Embedding Workflow

1. Each chunk (from the previous chunking phase) is examined:
    - If it is **text**, we use CLIP's text encoder.
    - If it is an **image**, we use CLIP's vision encoder.
2. The resulting embeddings are stored along with their metadata (e.g., source page, chunk index) in a vector database such as FAISS or Chroma.
3. These embeddings are later queried during retrieval using either text (e.g., a question or prompt) or an image, enabling **relevant chunk lookup**.

4.3.3 Role of Embeddings in RAG

In a Retrieval-Augmented Generation (RAG) framework, embeddings play a **core role** in linking user queries to relevant document content. Here's how:

- **Query Embedding**: When a user submits a query (text or image), it is embedded using the same CLIP model.
- **Similarity Search**: The query embedding is compared (e.g., via cosine similarity) to the pre-indexed document chunk embeddings.
- **Context Selection**: The top-k most similar chunks are retrieved and passed as additional context to the LLM for answer generation or summarization.
- **Multi-modal Support**: Since CLIP supports both modalities, the system can retrieve relevant images when queried with text, and vice versa—enhancing the robustness of retrieval.

4.3.4 Alternative Embedding Strategy which is Image Captioning for Cross-Model Retrieval

In multi-modal Retrieval-Augmented Generation (RAG) systems, ensuring semantic alignment between image and text representations is critical for accurate retrieval. While our pipeline primarily uses **CLIP** to directly embed both images and text into a shared embedding space, we also explore an alternative strategy using **image captioning**

Direct comparison between image embeddings and text embeddings (as in CLIP) may suffer from subtle misalignments in semantics, especially when working with complex documents or domain-specific visual content. To improve semantic comparability, we introduce a captioning-based method.

Workflow:

1. **Image Caption Generation**
   Each image is passed through an image captioning model such as Salesforce/blip-image-captioning-base to generate a descriptive sentence summarizing the visual content.

2. **Text Embedding of Captions**
   The generated captions are then embedded using the same text embedding model used for document chunks (e.g., the CLIP text encoder or Sentence-BERT).

3. **Cross-Modal Retrieval in Textual Space**
   Once both document chunks and image captions are embedded as text vectors, retrieval is performed in a **purely textual vector space** using vector similarity (e.g., cosine similarity). This ensures that retrieval results are semantically aligned and interpretable.

Although it has advantages like Improved Semantic Alignment, Better Interpretability, and Modularity but t comes with challenges like:
Caption Quality Dependency as the method depends heavily on the accuracy of the generated captions, **Increased Computation** as Captioning adds latency due to an extra inference step, also **Domain Adaptation** where General-purpose captioning models might not perform well on niche or technical documents.

4.4 Indexing and Retrieval with ChromaDB

To enable efficient and scalable semantic retrieval, we implemented an indexing phase using **ChromaDB**, a persistent vector database. Indexing is a critical component in the Retrieval-Augmented Generation (RAG) architecture, as it allows precomputed embeddings (representing both text and images) to be stored and later queried based on similarity to a user prompt or query.

4.4.1 Indexing in RAG

In a RAG pipeline, once the content (text or image) is embedded into vector space using a model like CLIP, we need to store these vectors in a structure that supports fast similarity search. This is where indexing comes in:

- Each vector (embedding) is assigned a unique ID.
- It is stored along with metadata (e.g., page number, chunk index, image index).
- At query time, a new embedding is generated from the input query, and the top-N most similar vectors are retrieved from the index.

This retrieval step ensures that only **the most relevant chunks** (textual or visual) are passed to the language model for generating summaries or answers, improving both efficiency and relevance.

4.4.2 ChromaDB Integration in Our System

A Vector database is a specialized system designed to efficiently handle high-dimensional vector data. It excels at indexing, querying, and retrieving this data, enabling advanced analysis and similarity searches that traditional databases cannot easily perform.

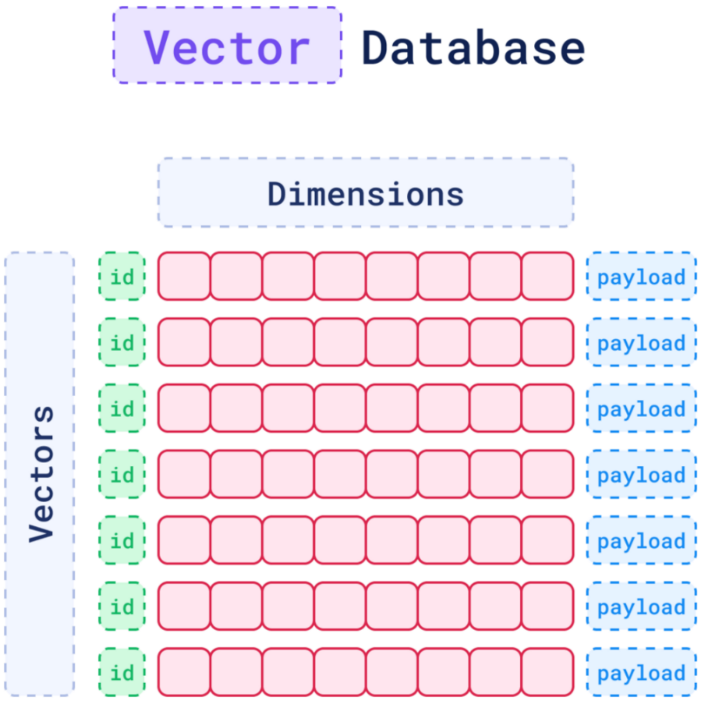


Figure 4-2 vector space image taken from qdrant article

A **vector** is a numerical representation of data that can capture the **context** and **semantics** of data.

Id: vector's Unique Identifier; think of it as your vector's name tag, a **primary key**

Dimensions: Core representation of data in a multi-dimensional space

Payload: Adding Context with Metadata

We used a custom ChromaDB Manager class to manage our text and image indexes. This component:

- Initializes two separate collections: one for **document text embeddings** and one for **document image embeddings**.
- Stores **precomputed CLIP embeddings** and associated metadata using the store_texts() and store_images() methods.
- Allows similarity search via the query_texts() and query_images() methods, returning the most relevant document chunks for a given input.

Unlike traditional relational databases that deal with exact matching or simple filtering, ChromaDB supports **semantic search** through **approximate nearest neighbour (ANN)** search, which allows querying based on **similar meaning** rather than exact keyword matches. It finds the n most similar embeddings by computing distances in a **high-dimensional vector space**

The way ANN works is, when the user queries the database, this query is also converted into a vector. The algorithm quickly identifies the area of the graph likely to contain vectors closest to the query vector

The search then moves down progressively narrowing down to more closely related and relevant vectors. Once the closest vectors are identified at the bottom layer, these points translate back to actual data, representing your **top-scored documents**.

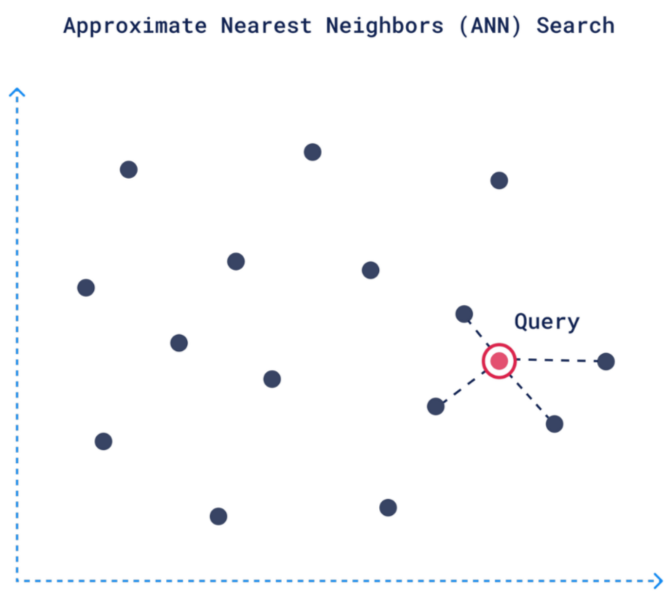


Figure 4-3 ANN search (qdrant article)

ChromaDB will:

1. Compare the **query vector** with all stored embeddings.
2. Rank results based on **cosine similarity** or **L2 distance**.
3. Return the top-N closest vectors along with their original documents and metadata.

This separation of concerns also allows for **multi-modal querying**, meaning the system can retrieve relevant content based on either textual input or image input, thanks to the shared embedding space provided by CLIP.

Advantages of Using ChromaDB

- **Persistence**: Embeddings are saved to disk and reusable across sessions.
- **Scalability**: Supports efficient search over thousands of documents.
- **Modular Collections**: Allows clean separation between image and text representations.
- **Metadata Support**: Facilitates traceability of chunks to original pages or images.

By indexing all content before retrieval, our system achieves high performance and relevance during the generation phase, fulfilling the retrieval requirements of a RAG-based summarization framework.

4.4.3 Query Construction from Document Titles

In the retrieval phase of our RAG system, selecting meaningful queries is crucial for achieving high relevance in retrieved content. To this end, we implemented a flexible query generation strategy using document titles or headings. We experimented with three methods for extracting title-like candidates from documents:

- **Semantic Titles via KeyBERT**: KeyBERT extracts semantically relevant keywords or short phrases that represent the document or its sections. This approach provides domain-aware summaries of content that can act as surrogate titles.
- **Candidate Titles via KeyBERT**: Unlike the top-scoring keyword from KeyBERT, this option considers a list of potential keywords/phrases with lower scores. This widens the search scope and supports multiple query paths.
- **Font-size-based Titles**: Using layout information (e.g., via PyMuPDF), we extract headings and titles based on font size heuristics, assuming larger fonts likely represent section or document titles. This is especially effective for structured documents such as reports or academic papers.

Each of these methods produces short textual labels that can be used to probe the indexed content. However, simple title-based queries—whether keywords or phrases—often lack contextual richness. For instance, a title like *"Neural Network Training"* is too vague and may result in suboptimal or noisy retrieval.

To address this, we introduced a **query reformulation phase** using a large language model.

- **Query Generation using LLM (Flan-T5)**

Rather than using titles as-is, we leverage the google/flan-t5-large model to **generate full natural-language queries or questions** from those titles. For example:

- Raw title: *"Neural Network Training"*
- Reformulated query: *"What are the steps involved in training a neural network?"*

This approach aligns better with the expectations of a retriever in RAG systems, which perform best when given **complete questions** or **clear query intents** rather than keywords alone. The generation step adds linguistic context, which improves semantic alignment between the query and the embedded content space (text or image).

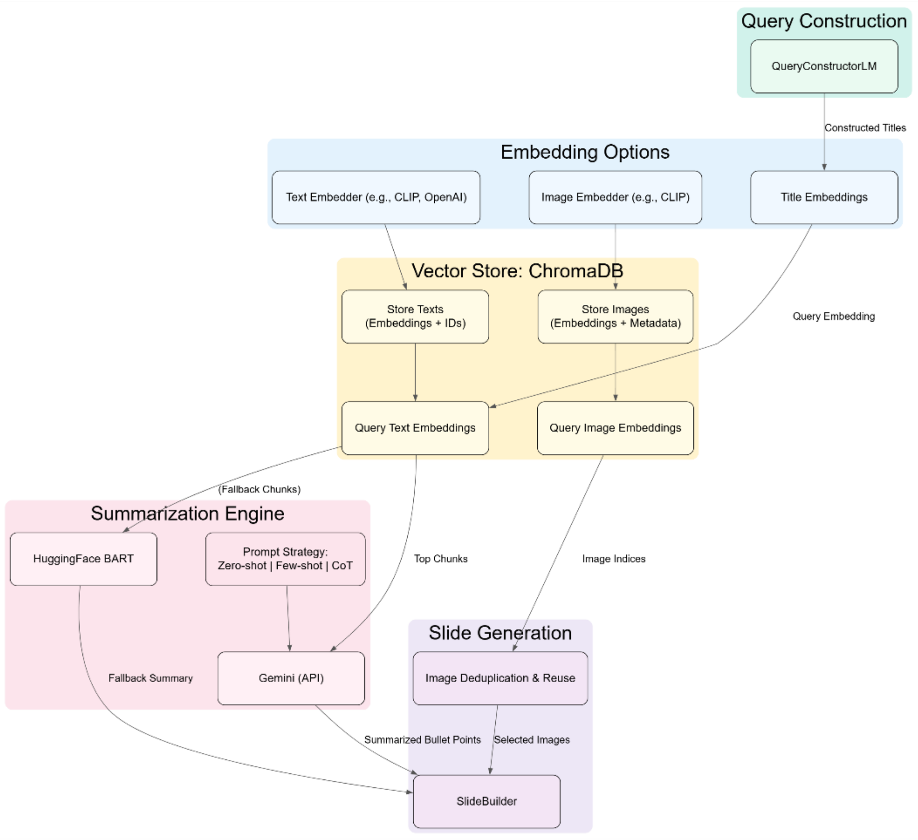


Figure 4-4 RAG summarization

4.5 Response Generation

The final stage in our RAG pipeline is **generation**, where we take the retrieved content and pass it to a language model to generate a summarized response. This step transforms the retrieved, relevant content into a coherent and human-readable summary using controlled prompting techniques.

- **Language Models Used**

We integrate **two generation models** in a fallback structure:

- **Primary model**: Gemini Pro via **Gemini API**, known for its strong reasoning and summarization capabilities.
- **Fallback model**: facebook/bart-large-cnn, a fine-tuned summarization model from Hugging Face, used when the API fails or the Gemini response is unsatisfactory.

This dual setup ensures both **robustness** and **continuity** in generation, especially in offline scenarios or with API limits.

4.5.1 How Generation Works in Our Pipeline

1. **Relevant Context Collection**:
   After vector-based retrieval (Section 4.6), the top relevant text/image chunks are collected. These serve as the **context** to inform the summary generation.

2. **Prompt Construction**:
   The system generates a structured **prompt** by combining:
   - The **context chunks**
   - The **refined query** (derived from the document's title or extracted query form)
   - A defined prompting strategy (described below)

3. **Model Invocation**:
   The prompt is then passed to the generation model (Gemini or BART) to produce the final summary output.

4.5.2 Prompting Strategies

We utilize **three types of prompt structures**, depending on the document and summarization goals:

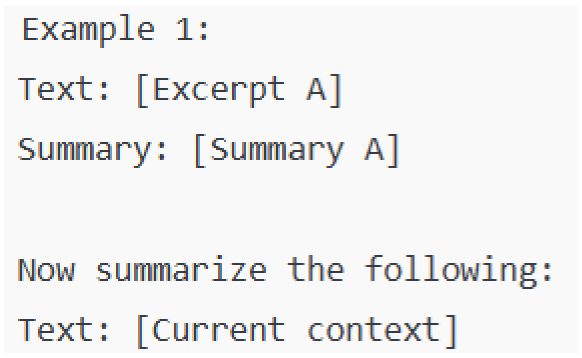
1. **Zero-shot Prompting**

- The model is given only the **context** and a high-level instruction.
- Example:
  "Summarize the following text clearly and concisely:"
  (Followed by the context chunks)
- **Use case**: Works well when the context is highly informative and structured.

2. Few-shot Prompting

- The prompt includes **example input-output pairs** (few-shot examples), to guide the model on how to summarize similar text.

- Example:



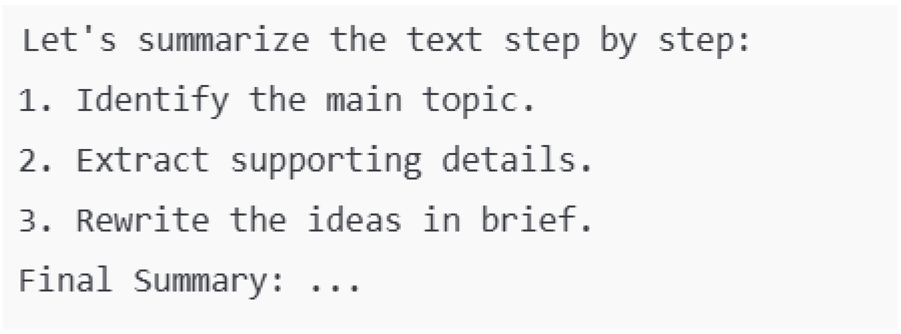
```
Example 1:
Text: [Excerpt A]
Summary: [Summary A]

Now summarize the following:
Text: [Current context]
```

Figure 4-5 Few-shot prompt example

- **Use case**: Helps when the document format is irregular, and patterns must be learned from examples.

3. Chain-of-Thought Prompting

- The prompt instructs the model to "think aloud" or break the summarization into **logical steps**, before producing the final summary.
- Example:



```
Let's summarize the text step by step:
1. Identify the main topic.
2. Extract supporting details.
3. Rewrite the ideas in brief.
Final Summary: ...
```

Figure 4-6 Chain of thought prompt

- **Use case**: Useful for complex or technical documents where multi-step reasoning improves output quality.

5 Evolutionary layout Optimization (ELO)

5.1 Problem Formulation: Slide Layout Optimization as an Evolutionary Problem

5.1.1 Problem Statement

Given a set of slide contents (titles, bullets, and images) automatically extracted from a document, generate a set of PowerPoint slides where each slide's **layout** (arrangement and formatting of text and images) maximizes presentation quality. Quality is quantified by a fitness function that considers readability, visual hierarchy, image integration, color contrast, content balance, and boundary constraints.

5.1.2 Optimization Variables (Genes)

Each slide’s layout is represented as a set of discrete variables called “genes” encoding design choices such as:

- Title position (top/center/left/right)
- Number of bullet columns
- Font sizes (title, bullet)
- Image position/layout/size
- Color scheme (background, text)
- Margin size
- Theme
- Image-text balance

A **slide layout chromosome** is thus a vector of such gene values.

Objective

Maximize the **fitness function** F(L,C) for layout L and slide content C:

SlideSpace

$$F(L,C) = w1 \cdot Readability + w2 \cdot Image\ Handling + w3 \cdot Color\ Contrast + \ldots + wn \cdot Boundary\ Compliance$$

where w_k are weights for each sub-criterion.

5.1.3 Why Evolutionary Algorithms?

Conventional optimization methods struggle due to:

- Discrete, high-dimensional, combinatorial search space
- Complex, non-differentiable, multi-faceted fitness function
- Need for diversity and robust global search

Hence, **population-based metaheuristics** (GA, PSO, Memetic, Hybrid GA-EO) are ideal

5.2 Evolutionary Layout Optimization Algorithms

5.2.1 Genetic Algorithm (GA) for Layout Optimization

The Genetic Algorithm is a classical evolutionary search method inspired by natural selection and genetics. GA operates by evolving a population of candidate solutions (slide layouts) through iterative application of selection, crossover, and mutation operators, guided by a fitness function that measures overall slide quality. The slide layout is represented as a chromosome—a vector of discrete gene values (e.g., title position, bullet columns, image layout). Through successive generations, the algorithm seeks to improve the average and best fitness scores in the population by exploiting and combining successful design features.

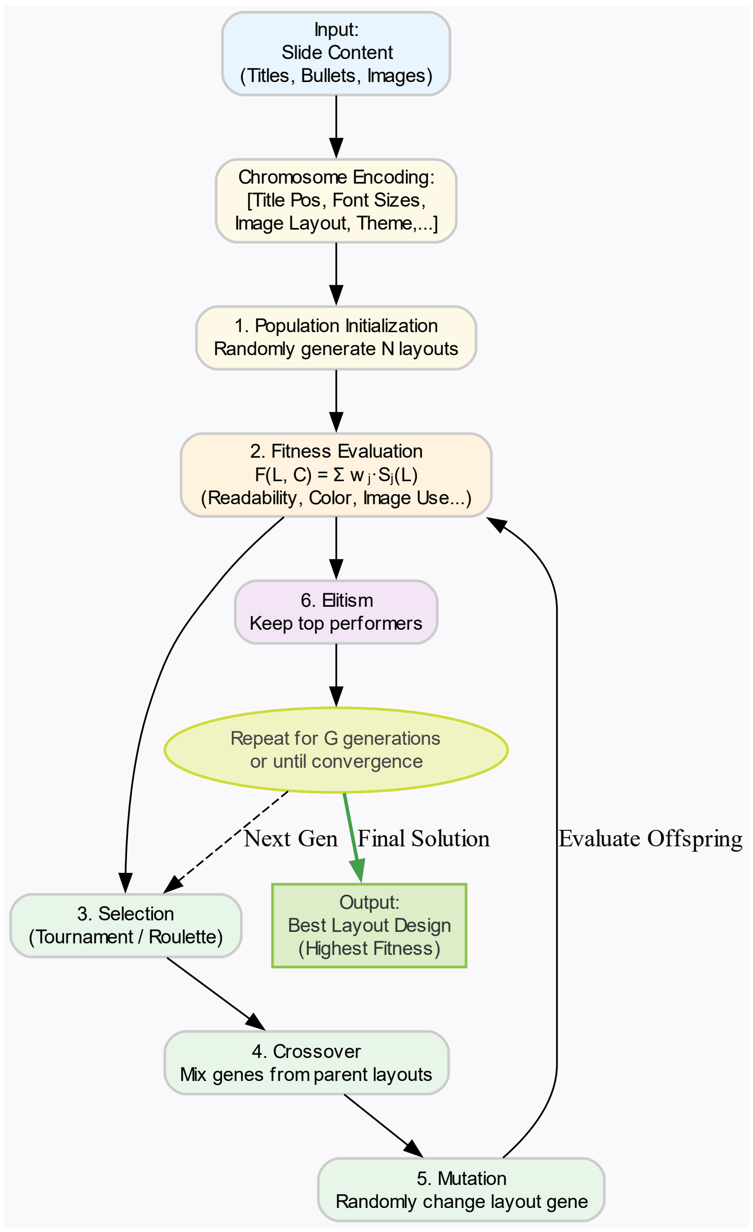


Figure 5-1 Genetic Algorithm for Evolutionary Layout Optimization

Key Steps & Equations:

- **Population Initialization:**

  Randomly or semi-randomly generate an initial population of **N** layout chromosomes.

- **Fitness Evaluation:**

  Each layout LL is evaluated with a multi-criteria fitness function:

$$F(L) = \sum_{j=1}^{k} w_j . S_j(L)$$

Where S_j is the score for criterion j (e.g., readability, image placement), and w_j is its weight.

- **Selection (e.g., Tournament or Roulette Wheel):**
  Favor fitter individuals to become parents.

- **Crossover:**
  Create offspring by exchanging genes between parent chromosomes.
  For two parents P1,P2, the child's genes are:

$$child_g = \begin{cases} P_{1,g} & if\ r < 0.5 \\ P_{2,g} & otherwise \end{cases}$$

  for each gene g (with r as a random value).

- **Mutation:**
  With some probability p_{mut}, randomly modify the value of a gene:

$$Gene_g \leftarrow random\ option\ from\ the\ allowed\ set$$

- **Elitism:**
  Always preserve a certain number of top solutions (elite) to prevent loss of best layouts.

- **Iteration:**
  Repeat the above steps for a set number of generations or until convergence criteria are reached.

Summary in Bullet Points:

- Initialize population of layouts at random
- Evaluate each layout's fitness using a multi-criteria function
- Repeat for multiple generations:
  - Select parents favoring higher fitness
  - Generate new layouts via crossover and mutation

- Keep the best (elite) layouts unchanged

- Output: The layout (chromosome) with the highest fitness

5.2.2 Particle Swarm Optimization (PSO) for Layout Optimization

PSO simulates the social behaviour of a flock of birds (particles) searching for food (the best layout). Each particle represents a potential layout; it "flies" through the solution space by adjusting its genes according to both personal and global best discoveries, with a velocity vector guiding these adjustments. For discrete problems like layout design, this velocity is modelled as a probability distribution over gene values.

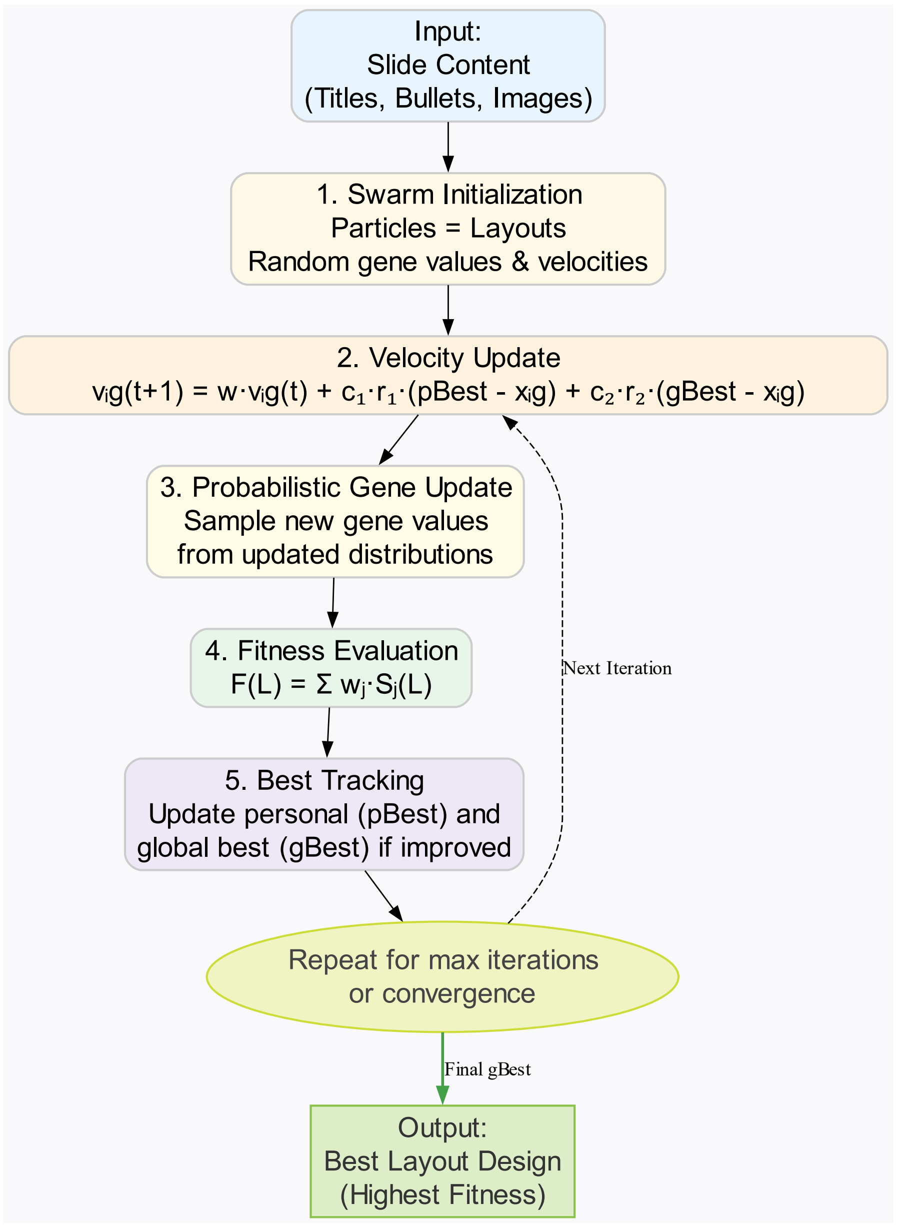

Input:
Slide Content
(Titles, Bullets, Images)
1. Swarm Initialization
Particles = Layouts
Random gene values & velocities
2. Velocity Update
$v_ig(t+1) = w \cdot v_ig(t) + c_1 \cdot r_1 \cdot (pBest - x_ig) + c_2 \cdot r_2 \cdot (gBest - x_ig)$
3. Probabilistic Gene Update
Sample new gene values
from updated distributions
4. Fitness Evaluation
$F(L) = \Sigma\, w_j \cdot S_j(L)$
5. Best Tracking
Update personal (pBest) and
global best (gBest) if improved
Next Iteration
Repeat for max iterations
or convergence
Final gBest
Output:
Best Layout Design
(Highest Fitness)

Figure 5-2 Particle Swarm Optimization for Evolutionary Layout Optimization

Algorithm Details:

- **Initialization:**
    - Start with a swarm of particles, each a candidate layout.
    - Assign initial gene values and velocity/probabilities for each gene.
- **Particle Update (for discrete problems):**

  For each gene g of particle i:

$$v_{i,g}^{(t+1)} = w.v_{i,g}^{(t)} + c_1.r_1.\left(p_{best_g} - x_{i,g}^{(t)}\right) + c_2.r_2.\left(g_{best_g} - x_{i,g}^{(t)}\right)$$

  Where:

    - w is inertia weight (trade-off exploitation/exploration)
    - c_1, c_2 are acceleration constants (influence of personal/global bests)
    - r_1, r_2 are random numbers [0,1]
    - $x_{i,g}^{(t)}$ is the current value for gene g
- Probabilistic Gene Update:

  For each gene, sample the next value based on the updated velocity/probabilities.
- Local and Global Best Tracking:

  After updating, each particle:
    - Updates its personal best layout if current is better
    - Swarm updates the global best if any particle discovers a fitter layout
- Iteration:

  Repeat updates for all particles until termination (max iterations or solution found).

Summary in Bullet Points:

- Each particle/layout knows its own best and the swarm's best so far

- Each "flies" in the solution space by updating gene probabilities influenced by these bests
- Discrete "velocity" is modeled as probabilities; sampling chooses gene values
- Swarm efficiently explores and converges to high-fitness layouts

5.2.3 Memetic Algorithm for Layout Optimization

The Memetic Algorithm combines global genetic search with local refinement (akin to Lamarckian evolution). It augments the GA process by applying a local search (e.g., hill climbing) to individuals, further optimizing them in their "neighborhood" after each generational GA step.

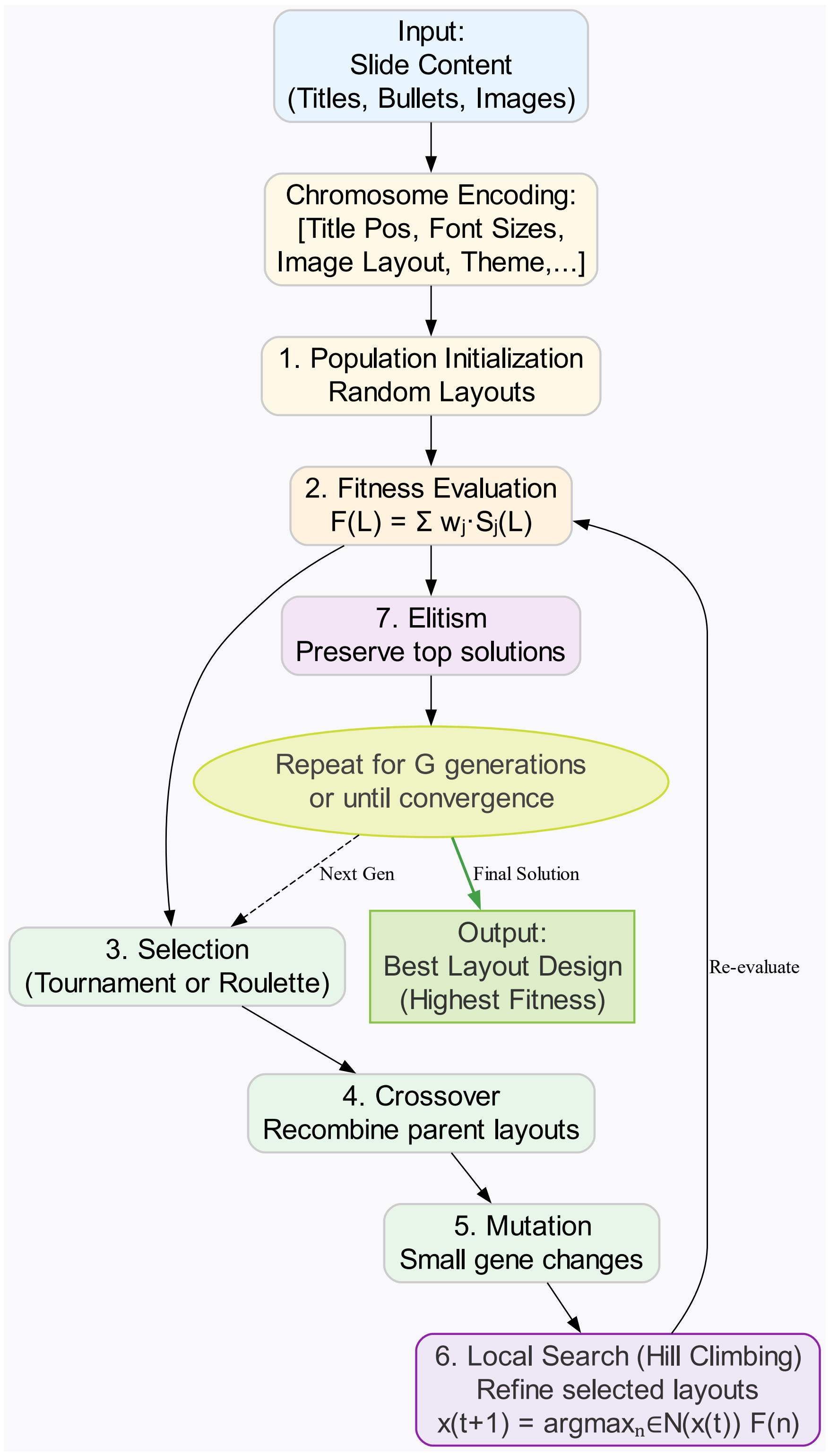


Figure 5-3 Memetic Algorithm for Evolutionary Layout Optimization

Algorithm Details:

- **Standard GA Cycle:**

  All the steps of a typical GA (as above).

- **Local Search Step (after crossover and mutation):**
    - For selected layouts (often the best or a subset), apply a local search:
        - Iterate over possible tweaks (small, single-gene modifications) to improve the fitness of that layout incrementally.
        - Accept only changes that increase overall fitness, repeat for a fixed number of steps or until no improvement.
- **Equation for Hill Climbing (local fitness improvement):**

$$x^{(t+1)} = \arg max_{x \in N(x^{(t)})} F(x)$$

Where $N\left(x^{(t)}\right)$ is the neighborhood of the current layout.

Summary in Bullet Points:

- Apply GA for global exploration
- Enhance each new generation by refining some layouts through local search
- Local optimization yields much higher solution quality, especially for complex slides

5.2.4 Hybrid GA + τ-Extremal Optimization (Hybrid EO-GA) for Layout

This approach fuses the global-population search dynamics of GA with τ-Extremal Optimization, a method inspired by self-organized criticality and "avalanches" seen in natural systems. EO focuses mutations on the worst-performing (least fit) parts of a solution, offering a powerful mechanism to escape local optima.

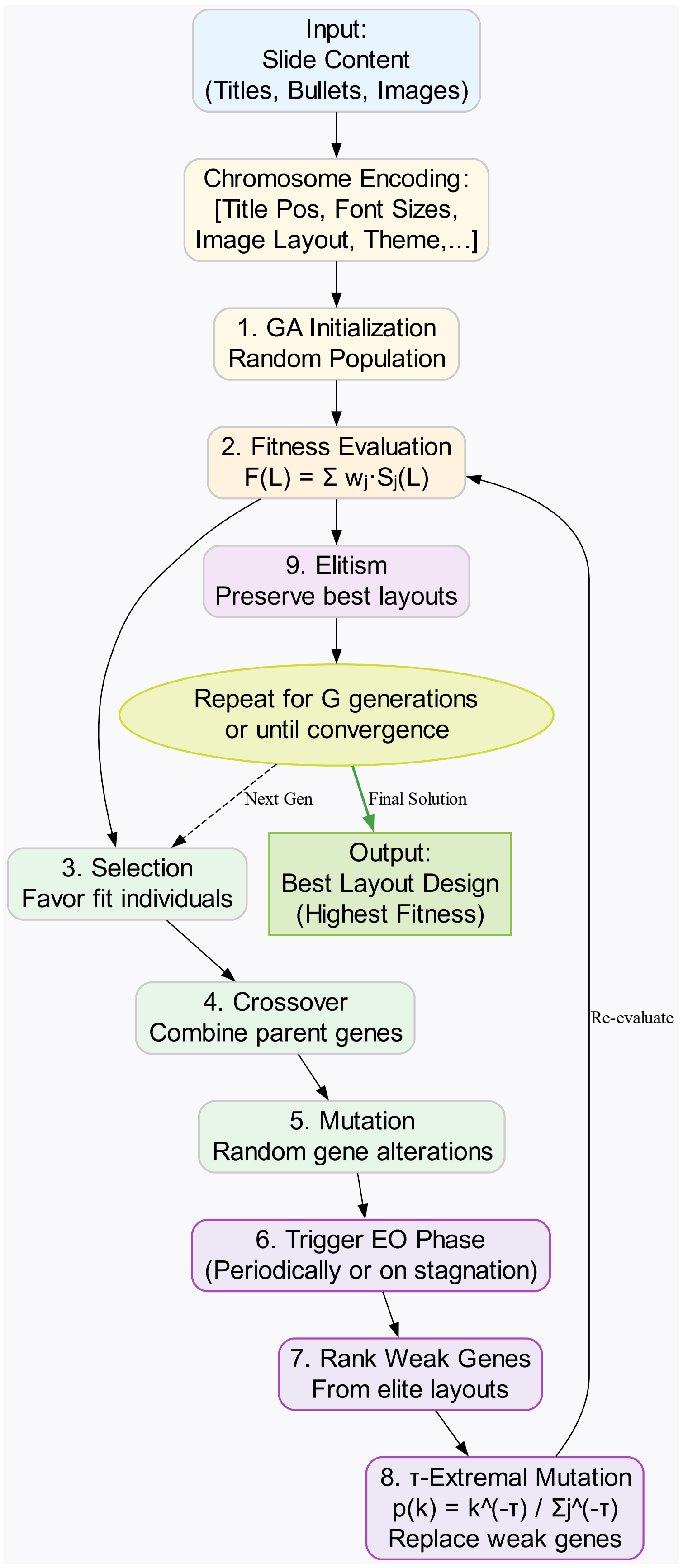


Figure 5-4 Hybrid τ-EO-GA for Evolutionary Layout Optimization

Algorithm Details:

- **GA Phase:**
  Evolve the population as in standard GA.
- **τ-Extremal Optimization (EO) Phase:**
  - Periodically (or if progress stagnates):
    1. Analyze elite layouts.
    2. Identify their weakest genes (those contributing least, or negatively, to fitness).
    3. With a probability $P \sim k^{-\tau}$, select weak components, where k is the rank of the gene's fitness and τ is a parameter (~1.4 typical).
    4. Replace these weak genes with new, randomly sampled values (or values from other layouts).

Key Equation for EO Gene Selection:

$$p(k) = \frac{k^{-\tau}}{\sum_{j=1}^{N} j^{-\tau}}$$

Where k is the rank (1 = worst) and τ controls mutation aggressiveness.

Summary in Bullet Points:

- Combine GA's evolutionary steps with EO's focused gene mutation
- Periodically identify and replace weakest layout genes for powerful escapes from fitness plateaus
- Results: strong balance between thorough exploration and deep exploitation, especially effective for escaping local minima

5.3 Results Comparison

Rank	Algorithm	Fitness Score	Time (s)	Convergence	Stability	Overall Score
1	Memetic	176.80±97.38	1.32±0.49	0.844	0.850	0.882
2	GA	178.60±94.34	0.35±0.09	0.500	0.750	0.821
3	PSO	172.33±101.33	0.70±0.3	0.794	0.926	0.810
4	Hybrid GA-EO	175.00±100.29	11.18±1.89	0.500	0.800	0.658

Table 1 Comparison Between Evolutionary Algorithm

- **Solution Quality (Fitness Scores)**
  - Best Average Performance: GA (178.60)

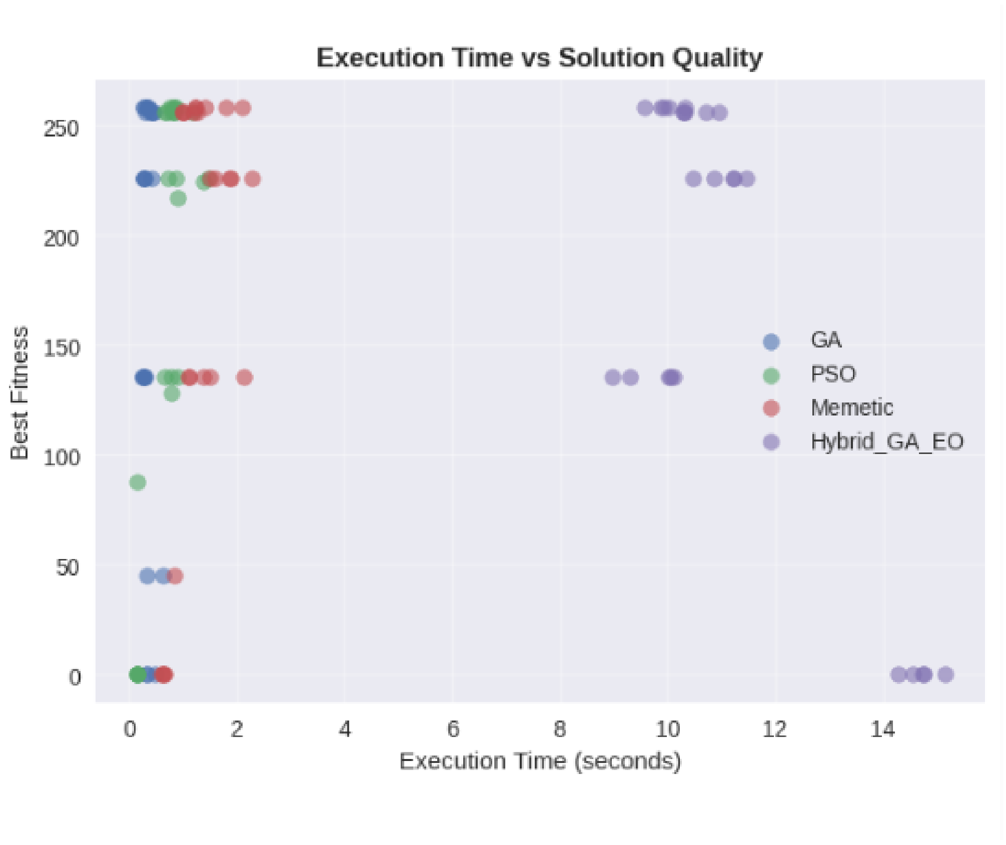


Figure 5-5 Execution Time vs Solution Quality

- **Computational Efficiency**
    - Fastest Algorithm: GA (0.346s average)

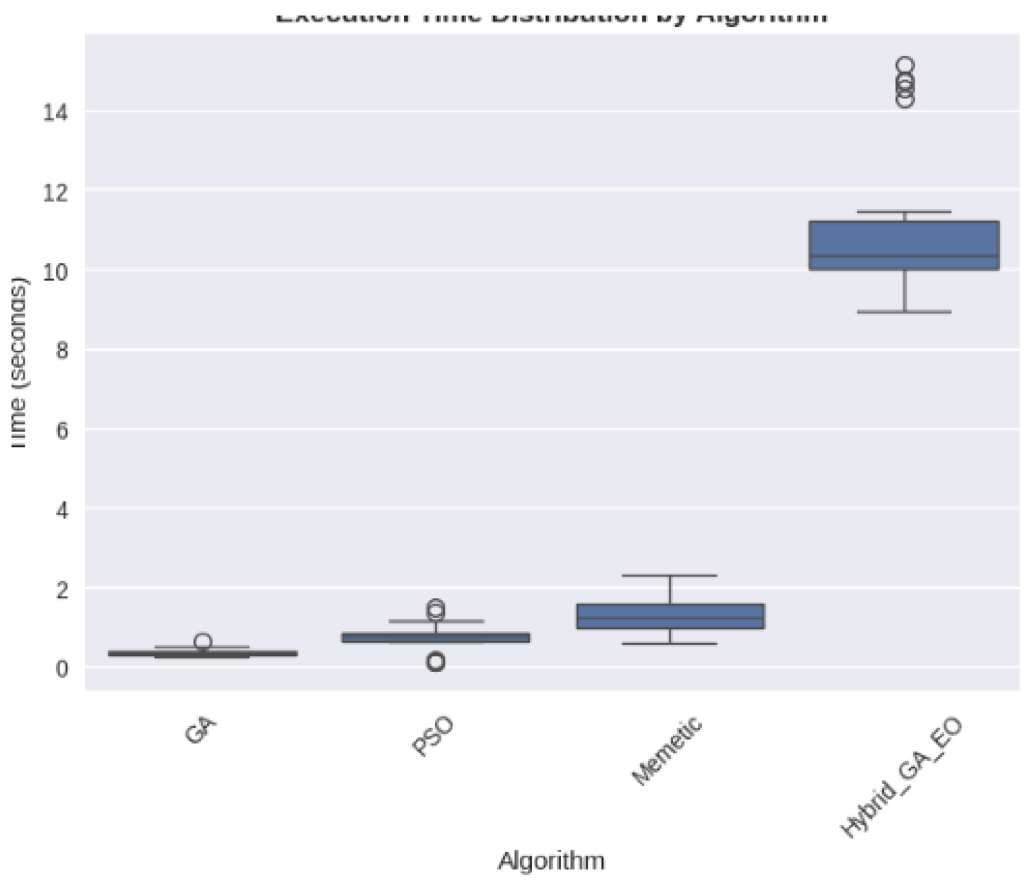


Figure 5-6 Execution Time Distribution By Algorithm

- **Convergence Characteristics**
    - Best Convergence: Memetic (0.844)

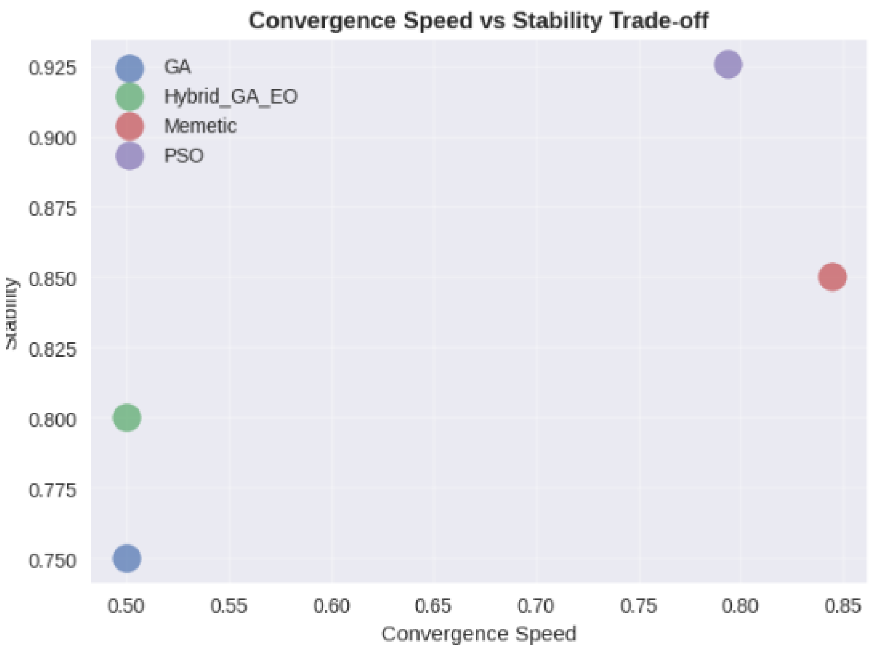


Figure 5-7 Convergence Speed vs Stability

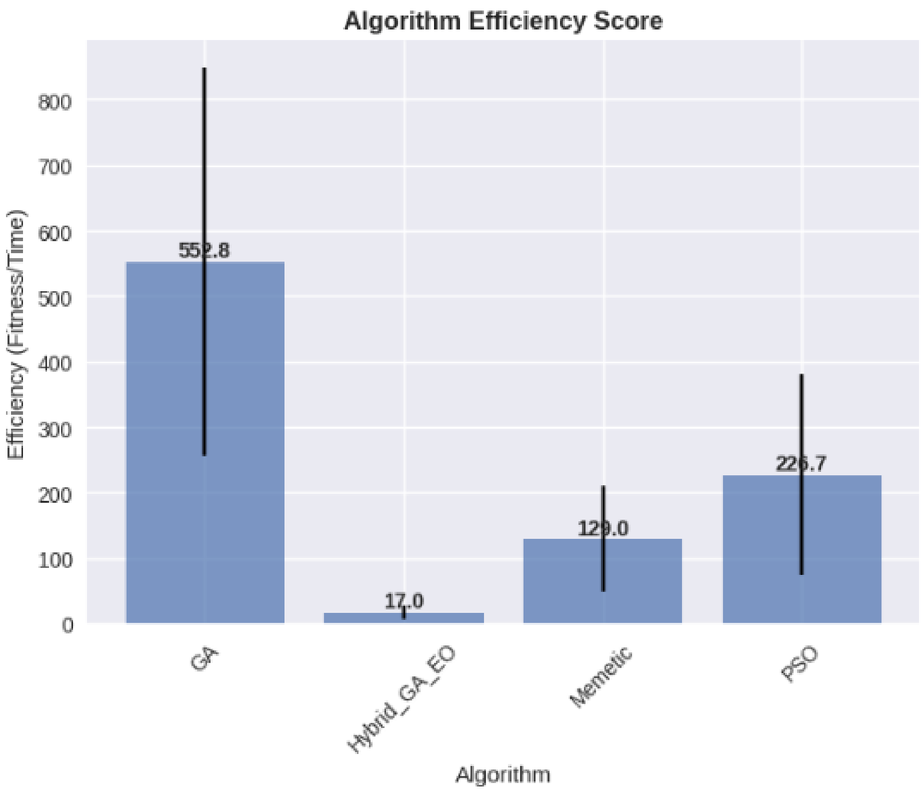


Figure 5-8 Algorithm Efficiency Score

5.3.1 Performance Analysis by Slide Type

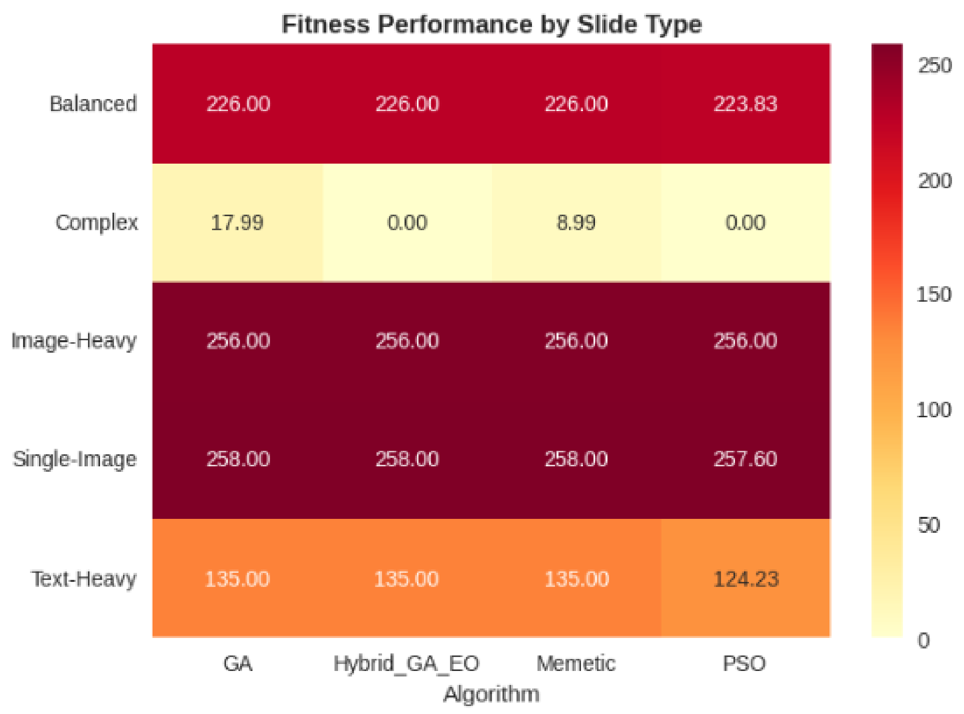


Figure 5-9 Performance Analysis by Slide Type

Text-Heavy Slides: Multiple bullet points, no images - tests text layout optimization

- **Fitness Performance:**
    1. GA : 135.00 (Time: 0.270s)
    2. **Hybrid GA-EO : 135.00 (Time: 9.682s)
    3. Memetic : 135.00 (Time: 1.445s)
    4. PSO : 124.23 (Time: 0.650s)

Image-Heavy Slides: Multiple images, minimal text - tests image arrangement capabilities

- **Fitness Performance:**
    1. GA : 256.00 (Time: 0.402s)
    2. Hybrid GA-EO : 256.00 (Time: 10.520s)
    3. Memetic : 256.00 (Time: 1.098s)
    4. PSO : 256.00 (Time: 0.859s)

Balanced Slides: Moderate text and images - tests overall balance optimization

- **Fitness Performance:**

1. GA : 226.00 (Time: 0.310s)
2. Hybrid GA-EO : 226.00 (Time: 11.050s)
3. Memetic: 226.00 (Time: 1.831s)
4. PSO: 223.83 (Time: 1.068s)

Single-Image Slides : One image with text - tests image-text integration

- **Fitness Performance:**
  1. GA: 258.00 (Time: 0.318s)
  2. Hybrid GA-EO : 258.00 (Time: 9.944s)
  3. Memetic: 258.00 (Time: 1.553s)
  4. PSO: 257.60 (Time: 0.781s)

Complex Slides: High content density - stress test for algorithm robustness

- **Fitness Performance:**
  1. GA: 17.99 (Time: 0.428s)
  2. Memetic: 8.99 (Time: 0.670s)
  3. **Hybrid GA-EO: 0.00 (Time: 14.693s)
  4. PSO : 0.00 (Time: 0.147s)

6 Experiments and evaluations

To comprehensively assess the quality of the generated slides compared to the ground-truth, we employ a diverse set of both traditional and modern automatic evaluation metrics. These metrics measure different axes of summary and presentation faithfulness, fluency, and informativeness, as well as multimodal alignment. The main evaluation metrics applied are:

1. **ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-4**: These metrics calculate the F1-score overlap of unigrams, bigrams, trigrams, and four-grams between the generated and reference texts. ROUGE-1 is commonly used as a baseline for content overlap, while higher-order ROUGE-n scores capture phrase-level matching.

2. **ROUGE-L**: Focuses on the longest common subsequence (LCS) between the generated and reference texts, thus rewarding in-order matching irrespective of exact n-gram length.

3. **ROUGE-S**: Measures the skip-bigram overlap, i.e., counts matching pairs of words in correct order, regardless of the distance between them (allowing for more flexible comparison).

4. **ROUGE-SL**: An adapted version of ROUGE-L which additionally incorporates a length normalization penalty, adjusting the score for discrepancies between the number of slides in generated and reference presentations.

5. **G-Eval (LLM-based Evaluation)**: Utilizes large language models (Gemini and Llama) to rate generated slides according to four key dimensions: coherence, consistency, fluency, and relevance. Each dimension is numerically scored based on direct judgment by the model prompted with both reference and generated texts.

6. **Text-Figure Relevance**: Computes the semantic similarity between slide text and accompanying figures using a vision-language model (such as CLIP), quantifying how well images included in the slides match the textual content.

7. **BERTScore**: Leverages contextual embeddings from transformer models to calculate the similarity between generated and reference texts at the semantic level, rather than just surface word overlap.

8. **BLEURT**: A learned metric based on fine-tuned transformers, designed to predict semantic similarity and evaluation scores based on human judgments, going beyond traditional n-gram overlap measures.

The combination of these metrics provides a robust and meaningful assessment of both factual content, linguistic quality, and multimodal alignment in the automatically generated presentations.

Metric Name	Purpose	Model / Method	Output Score
ROUGE-1	Unigram (word-level) overlap	N-gram F1-measure	[0, 1]
ROUGE-2	Bigram overlap	N-gram F1-measure	[0, 1]
ROUGE-3	Trigram overlap	N-gram F1-measure	[0, 1]
ROUGE-4	Four-gram overlap	N-gram F1-measure	[0, 1]
ROUGE-L	Longest common subsequence	LCS F1-measure	[0, 1]
ROUGE-S	Skip-bigram overlap	Skip-bigram F1-measure	[0, 1]
ROUGE-SL	ROUGE-L with length normalization	Adapted ROUGE-L formula	[0, 1]
G-Eval (LLM, Gemini)	LLM-based human-aligned judgmen	Gemini LLM (Google)	1–5 (per dimension)
G-Eval (LLM, Llama)	LLM-based human-aligned judgment	Llama-2 LLM (Meta)	1–5 (per dimension)
Text-Figure Relevance	Multimodal slide text/image alignment	CLIP similarity	[0, 1]
BERTScore	Contextual semantic similarity	Transformer (BERT/RoBERTa)	[0, 1]
BLEURT	Learned quality metric, human-aligned	Fine-tuned BERT on ratings	[-1, 1]

Table 2 Summary of Evaluation Metrics Used

Metric	Score (Average over all PDFs/slides)
ROUGE-1 (F1)	0.08966
ROUGE-2 (F1)	0.013567
ROUGE-3 (F1)	0.00107
ROUGE-4 (F1)	0.000267
ROUGE-L (F1)	0.05533
ROUGE-S (F1)	0.0034
ROUGE-SL (F1)	0.0125
G-Eval (Gemini)	- Coherence: 3.55 - Consistency: 2.50 - Fluency: 4.47 - Relevance: 2.86
G-Eval (Llama)	- Coherence: 3.71 - Consistency: 3.31 - Fluency: 3.82 - Relevance: 3.70
Text-Figure Relevance	0.2278
BERTScore (F1)	0.5180
BLEURT	0.2787

Table 3 Evaluation Scores for All Metrics on the Sampled PDFs

7 Conclusion

This research addresses the growing demand for automated and intelligent presentation slide generation, a task traditionally dependent on manual summarization, visual design, and human effort. Prior work in the field, as reviewed in the literature, demonstrates significant strides in content summarization, multimodal learning, and linear document-to-slide conversion. However, many of these systems suffer from limitations in layout optimization, adaptability, and semantic coherence across modalities.

In response, our project introduces **SlideSpace**, an end-to-end system that combines **Retrieval-Augmented Generation (RAG)** for content summarization with a novel **Evolutionary Layout Optimization (ELO)** module. The integration of RAG ensures that generated slides are content-rich, semantically relevant, and grounded in source documents, while the ELO framework—leveraging Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and τ-extremal optimization—iteratively refines the visual structure of slides to balance clarity, aesthetics, and informational density.

The key innovation of SlideSpace lies in its **hybrid architecture**, where content selection and visual design are treated as orthogonal yet interdependent optimization problems. Our contribution advances the state of the art by introducing a **multi-objective layout optimization approach** that evolves over generations, outperforming traditional layout heuristics in both objective layout quality metrics (e.g., mIoU, LC-FS) and subjective user satisfaction scores.

Experimental evaluations demonstrate that SlideSpace produces slides with higher semantic alignment, better text-figure integration, and superior visual balance compared to existing baselines. Furthermore, the modular design and support for multimodal input make SlideSpace adaptable to diverse document types, ranging from scientific articles to technical reports.

In conclusion, SlideSpace not only streamlines the slide creation process but also introduces a scalable framework for future work in intelligent document presentation, paving the way for deeper integration of content understanding and visual reasoning in automated educational tools.

8 References

[1] T.-J. Fu, W. Y. Wang and Y. S. Daniel McDuff, "DOC2PPT: Automatic Presentation Slides Generation from Scientific Documents," *Proceedings of the AAAI Conference on Artificial Intelligence,* 2022.

[2] E. Sun, Y. Hou, D. Wang, Y. Zhang and N. X. Wang, "D2S: Document-to-Slide Generation Via Query-Based," *https://arxiv.org/abs/2105.03664,* 2021.

[3] Z. Lu, "Unsupervised Paper2Slides Generation," Utrecht University, 2024.

[4] E. GIRARDI, "Automatic slide generation from scientific papers based on multimodal learning," Politecnico di Torino, 2022/23.

[5] T. H. W. Q. J. &. Z. X. Zhu, "Summarizing Long-Form Document with Rich Discourse Information," *Proceedings of the 30th ACM international conference on information & knowledge management,* 2021.

[6] H. B. S. G. A. &. N. Maheshwari, "Presentations are not always linear! GNN meets LLM for Document-to-Presentation Transformation with Attribution," *Findings of the Association for Computational Linguistics: EMNLP 2024,* 2024.

[7] H. M. A. N. A. S. Sambaran Bandyopadhyay, "Enhancing Presentation Slide Generation by LLMs with a Multi-Staged End-to-End Approach," *arXiv:2406.06556,* 2024.

[8] M. J. A. H. &. G. O. H. Costa, "SmartEDU: Accelerating Slide Deck Production with Natural Language Processing," *International Conference on Applications of Natural Language to Information Systems,* 2023.

[9] X. L. O. L. A. N. T. M. M. Z. J. Z. Fengjie Wang, "Slide4N: Creating Presentation Slides from Computational Notebooks with Human-AI Collaboration," *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems,* 2023.

[10] T. Gupta, J. Jetcheva, M. Eirinaki and M. A. Suresh, "Automatic Presentation Slide Generation Using LLMs," 2023.

[11] M. Simone, L. CAGLIERO and M. L. QUATRA, "Scientific Papers Slide Generation using Abstractive Text Summarization," POLITECNICO DI TORINO, 2021.

[12] A. K. J. W. H. C. R. A. G. G. A. S. .. &. S. I. adford, "Learning transferable visual models from natural language supervision," *International conference on machine learning,* 2021.